LaTeX

桑大勇　王瑛

科技文献排版系统 LaTeX 入门与提高

Keji Wenxian Paiban Xitong LaTeX Rumen Yu Tigao

Keji Wenxian Paiban Xitong LaTeX Rumen Yu Tigao

- 国际权威学术机构指定排版交流格式
- 符号公式随心所欲
- 支持多种操作系统平台
- 易于转换成 PostScript 或 pdf 文档
- 便于扩充并形成个性化的排版环境

TeX

武汉大学出版社

科技文献排版系统 LaTeX 入门与提高

Keji Wenxian Paiban Xitong LaTeX Rumen Yu Tigao

Keji Wenxian Paiban Xitong LaTeX Rumen Yu Tigao

ISBN 7-307-03145-0

9 787307 031456

责任编辑 夏炽元
责任校对 李桂珍
版式设计 支 笛
装帧设计 莺 子

ISBN 7-307-03145-0/TS·10
定价:17.50 元

科技文献排版系统
LaTeX 入门与提高

桑大勇　王　瑛

武 汉 大 学 出 版 社

图书在版编目(CIP)数据

科技文献排版系统 LaTex 入门与提高/桑大勇,王　瑛.—武汉：武汉大学出版社,2001.5
ISBN 7-307-03145-0

Ⅰ.科…　Ⅱ.①桑…　②王…　Ⅲ.排版—应用软件 LaTex　Ⅳ.TS803.23

中国版本图书馆 CIP 数据核字(2000)第 59619 号

责任编辑：夏炽元　　责任校对：李桂珍　　版式设计：支　笛

出版：**武汉大学出版社**　(430072　武昌　珞珈山)
(电子邮件：wdp4@whu.edu.cn　网址：www.wdp.whu.edu.cn)
发行：新华书店湖北发行所
印刷：湖北省通山县印刷厂
开本：787×1092　1/16　　印张：11.625　　字数：277 千字
版次：2001 年 5 月第 1 版　　2001 年 5 月第 1 次印刷
ISBN 7-307-03145-0/TS·10　　定价：17.50 元

内 容 提 要

LaTeX 是一种与操作系统无关的排版注记语言，被许多国际权威学术机构指定用于提交文档的描述。本书旨在使读者全面掌握其最新标准 LaTeX2e，其中第一章介绍 LaTeX 排版的主要特色和输入文档的基本结构；第二章深入讨论文字文档的排版，重点介绍 LaTeX 中最主要的文字排版命令和环境；第三章介绍一些辅助文字文档排版的功能；第四章介绍数学公式的排版方法；第五章介绍美国数学协会(AMS)扩充的、已经移植到 LaTeX 标准当中的 AMS-TeX 标准的数学排版命令；第六章介绍 LaTeX2e 的图形和表格排版环境；第七章介绍 LaTeX 中 PostScript 图像的插入以及文档颜色的处理；第八章介绍 LaTeX2e 中用户自定义类包文件的设计；第九章介绍常见的几种 LaTeX 排版环境；最后的附录部分将几乎所有的 LaTeX2e 排版命令加以简要说明并标注其在书中的页码。本书适合于 Windows 用户中所有需要同国外进行学术交流的科技工作者，并同样适合于那些不使用 Windows 或 Microsoft Word 而又需要进行印刷级文档排版的各类人员。

前　言

在当今时代，计算机的一个最通用的功能就是使用字处理系统对文本进行电子化处理，这种处理工作主要由如下四个步骤完成：文本输入到计算机里，存储起来供以后修改、扩充和删减；格式化输入文本（格式化后的文本我们姑且称之为文档），使其以相同长度的行和特定尺寸的页显示出来；在计算机的监视器上显示格式化后的结果；把最终的输出送到打印机上打印出来。有很多字处理系统可以在一个软件包中同时实现这四方面的功能，因此用户也就意识不到上面这几种划分，这种字处理系统我们称之为排版系统。我们大部分读者可能都使用过这样或那样的排版系统，但经常令我们苦恼而又无奈的是，一个排版软件所排好的文档往往不能被另一个排版系统所识别。

随着科技的飞速发展和 Internet 的迅猛普及，中国广大的科技工作者正面临前所未有的机遇与挑战。一方面我们能够及时获取全世界同行的最新研究成果，把握本学科最前沿的科研方向；另一方面，为了使中国成为科技大国，我们必须做出与之相应的出色的研究成果，而更为重要的事还是要把这些成果尽快地向世界公布。国外的学术杂志、专业学术会议以及各个出版机构早就接受电子稿件，有时为了加快学术交流的速度甚至要求必须用电子邮件投稿，这样排版软件之间的互不兼容将是一个极大的障碍。为此，我们必须尽量使用一种国际上大都能够接受的排版格式，才能更好地拥有在世界科技舞台上的发言权。

国内用户在撰写文稿时大都使用 Microsoft 的 Word，但是 Word 至少存在下面几个不足：(1) 受排版软件版本的限制，Word 中文版与西文版、低版本与高版本之间均存在某些不完全兼容的格式问题；(2) 受操作系统的限制，要求排版者和文档接收者都必须使用同样的操作系统和同样版本的 Word，而 Word 的排版效果大多数专业出版机构都不能满意；(3) Word 在使用单独的工具编排数学公式时，必须考虑公式的位置、公式中字符的大小和类型等等，而且排版的效果不佳，更不用说缺乏对定理排版和公式等的自动编号和引用等能力。LaTeX 作为一种具有国际标准、支持多种平台的排版技术，可以解决上述所有这些（以及许多没有罗列出来的）问题，因此理应得到我们重视。作者受到自身水平的限制，不可能在本书中将 LaTeX 的所有特性都阐述出来，相信只要您使用 LaTeX，就会比作者得到更多的发现和惊喜。

本书适用于 Windows 用户中所有需要同国外进行学术交流的科技工作者，并同样适合于那些不使用 Windows 或 Microsoft Word 而又需要进行印刷级文档排版的各类人员。

非常感谢武汉大学出版社给我们提供一个分享 LaTeX 果实的机会，同时感谢软件工程国家重点实验室的许多老师在本书成稿过程中对第一作者的关心和帮助。

作　者

2000 年 11 月

目 录

第一章 LaTeX 概貌

1.1 LaTeX 排版系统简介

1.1.1 LaTeX 系统发展过程

LaTeX 系统从最初的纯 TeX(Plain TeX)系统，经历了 LaTeX2.09 版发展到今天的 LaTeX2e 标准。有一个国际化的工作组正在实施 LaTeX 项目，希望在不久的未来能发展到 LaTeX3 版本。LaTeX 系统版本进化速度之所以如此缓慢，作者的理解是为了保护用户的利益，除非内容发生了极大甚至是根本性的变化，不会像现在的某些软件一样动辄就到了第十几二十版。当然许多新的功能也在不断地向现行版本中添加，为此 LaTeX 在保证新内容不会引起不兼容问题的前提下，使用版本日期来确定文档可能使用的最新功能。本书将以 LaTeX2e 版为基础进行介绍，所有内容若未特别指出，均属该版本所支持的范畴。

1. TeX 系统

TeX 是由 Donald E. Knuth 书写的一个计算机程序，目的是对文本和数学公式进行排版。TeX 的发音应该是“Tech”，其中“ch”的发音同其在德语单词“Ach”或苏格兰语单词“Loch”中的发音。最基本的 TeX 程序是由一些很基本的命令组成的，它们可以完成简单的排版操作和程序设计功能。TeX 也允许用这些基本命令定义一些更复杂的高级命令，这样就可以利用低级的结构块，形成一个用户界面相当友好的环境。当处理器运行 TeX 时，该程序首先读取所谓的格式文件，格式文件中包含各种以基本语言写成的高级命令，也包含分割单词的连字号安排式样。接着处理程序就处理源文件，源文件由要处理的真正文本以及在格式文件中已定义了的各种命令（保留字）组成。

Knuth还设计了一个名叫Plain TeX的基本格式，以便与低层次的TeX互应。Plain TeX格式依然是TeX字处理的相当基本的部分，以致于我们有时候根本分不清到底哪是真正的TeX处理程序，哪是Plain TeX格式。大多数声称只使用TeX的人，实际上指的是只用Plain TeX，因此人们经常把TeX和Plain TeX认为是同一事物。

2. LaTeX 系统

Plain TeX 的重点还只是停留在如何排版的层次上，对 TeX 深层功能的进一步发掘需要相当高超的编程技巧，因此它的应用面向高级排版和程序设计人员。正是由于这种原因，美国计算机学家 Leslie Lamport 开发了 LaTeX 格式，这种格式提供了一组生成复杂文档所需要的更高级命令。利用这种格式，即使使用者没有排版和程序设计的知识也可以充分发挥由 TeX 所提供的强大功能，能在几天甚至几小时内生成大量具有书籍印刷质量的结果。在生成复杂表格和数学公式方面，这一点表现得尤为突出。

LaTeX 相对于 Plain TeX 而言，更像一个包装语言。它可以在作者根本不知道所以然的情况下，自动给出标题、章节、表格目录、交叉引用、公式编号、文献引用、浮动图表

等等。版面布局信息包含在类(class)文件中，这些类文件并不是位于源文件中的。这些布局可以改动，也可以直接套用。LaTeX 是在 20 世纪 80 年代出现的，周期性地进行更新和修订，经过了很多年以后其版本号固定为 2.09，以后的修订只是用日期来区分。

3. LaTeX2e 系统

由于LaTeX的普及及其在许多原本没有考虑到的领域中的扩展，同时由于计算机技术的日新月异（特别是价格越来越低廉）以及功能强大的激光打印机的出现，使得相当广泛的一类排版格式都冠以LaTeX的标签。为了再次统一自LaTeX2.09版本以后所进行的各种修补工作并建立一个真正的、能够满足较长时期内排版需求的LaTeX改进标准，在Frank Mittelbach的领导下，Leslie Lamport, Chris Rowley和Rainer Schopf创立了LaTeX3项目组。他们的长期目标是得到LaTeX的一个新版本3，在该版本中建立一个最优的、有效的命令集合。1994年他们发行了一个新的版本，为了同老版本LaTeX2.09相区别，新版本命名为LaTeX2e。

实际上在LaTeX2e出现之前，其处理字体安装和选择的一些部分已经以新字体选择框架(NFSS)的形式公开并被许多组织或个人集成到其排版软件中。NFSS有两个原本并不兼容的版本，两个版本分别对应于LaTeX2.09和LaTeX2e，后来LaTeX3项目组以一种完全与2.09版本兼容的方式对NFSS进行了重新实现。

1.1.2 LaTeX 排版的主要特点

在前言中我们列举了拥有广大用户群体的 Microsoft Word 系统所具有的缺陷，在此不再赘述。而 LaTeX 排版系统至少存在下面这些显著特点，读者可以将它们同现在您所使用的排版软件进行比较。囿于作者对 LaTeX 的理解程度，肯定尚有许多特色被遗漏。

1. 国际权威学术机构指定排版格式

国际上许多权威学术机构都将 LaTeX 排版格式作为标准的文档格式。各种数学刊物、国际数学会议上被定为标准的论文投稿、编排软件，例如著名的国际数学刊物《Journal of Group Theory》(《群论杂志》)就将 LaTeX 文件定为标准论文投稿格式，美国数学协会(AMS)甚至将它所有的会刊论文格式都定为 LaTeX。

2. 符号公式随心所欲

Knuth 设计了另一个软件 METAFONT，用来生成各种字符字体，在标准的 TeX 软件包中有 75 种不同设计尺寸的字体，而且每种字体有八种不同的放缩比例。为了满足其他应用的需要，TeX 还设计了其他字符字体，如日耳曼、希腊、古斯拉夫语或日语等等字母的字体，有这些字母的文本也可以用书籍质量排版出来。LaTeX 软件对 TeX 进行了发扬光大，具有更为强大的数学公式、符号排印能力，它提供了方便的宏命令，将复杂的公式系统化，可以轻松地进行各种符号、文字、公式的编排，在公式编辑方式下可以编辑所有的数学公式。美国数学协会在 LaTeX 的基础上针对数学公式的排版问题专门进行扩充并发布 AMS-LaTeX 软件包，现已被接受为 LaTeX 标准。LaTeX 系统将数学公式的印刷看作不可分割的一个重要组成部分，而诸如 Microsoft Word 等许多排版系统只是额外附加一个数学公式工具，比如 Word 只能将公式作为一种外部对象插入文档，不仅使得在没有安装公式工具的 Word 中无法编辑（实际上作者多次遇到过这种情况：以前保存的带公式文档再次

在同一个 Word 环境下打开后也无法对其中的公式进行编辑），而且公式与文档中其他元素间无法很好协调（如字体字号、对齐间隔、确省设置等）。在 LaTeX 中公式和文档的其他部分可以更紧密、更协调地进行排版，也正因为如此，LaTeX 被认为是目前国际上排印数学公式和科学符号能力最强的软件，深受广大科技人员的欢迎。

3. 文档易于网上传输

LaTeX 文档采用一种文本解释方式来表示文档的开始、结束和符号，例如 "\alpha" 在 LaTeX 中打印效果是小写希腊字符 "α"；LaTeX 文件在文档中只包含 ASCII 扩展字符集前 128 位（和 TXT 文件相同），文档输入文件同其他系统的文件相比尺寸较小，非常有利于在 Internet 上通过电子邮件传输。不论在 Internet 作为最为快捷的交流通道的今天，还是在无纸印刷时代的明天，这种特性无疑顺应了技术发展的潮流。

4. 支持多种操作系统平台

Word 迄今为止也只能在 Microsoft 的 Windows 系列操作系统上运行，也就是说接受 Word 电子文稿的机构必须同样使用 Microsoft 的操作系统加上同样版本的 Word 进行排版印刷，而这一点往往并不容易满足。随着台式机性能的飞速提高，在个人机上使用 unix 类或其他操作系统的用户越来越多，他们对排版工具的需求日益迫切，而 LaTeX 排版系统可以让你轻松跨越这道操作系统屏障，因为各种平台上几乎都存在 LaTeX 系统。假设很不幸你找不到在你的机子上运行的 LaTeX 软件，但是只要使用任何文本编辑程序，按照本书中介绍的命令和文档格式输入文档源文件、然后发送给文档接收者就可以了。为了帮助您找到合适的软件，这里给您提供一个很好的 TeX 国内下载网站：

http://octane.math.ustc.edu.cn/~texguru/

5. 易于转换成 PostScript 或 pdf 文档

LaTeX 排版时在文档中可以插入 PostScript 的 eps 格式的图像，同时系统生成的 dvi 文件很容易转换成 PostScript 的 ps 格式或 Acrobat 的 pdf 格式。ps 文件和 pdf 文档在国外拥有广泛的用户群，近年来在国内的用户也越来越多，尤其在网络上发布的文档中它们扮演着重要的角色。

6. 便于扩充并形成个性化的排版环境

在编排文档时，我们总会发现有些地方基本的 LaTeX 功能不够用，比如文档中想包含图形、彩色文本或者一段来自某个文件的源代码，此时就得加强 LaTeX 的能力，这种加强在 LaTeX 中用包来实现。LaTeX 本身随产品发布了许多可选的包文件，用户可以自己取舍并在文档开头进行声明(参见 1.2.2 节)，就可以使用符合自己需求的排版功能。另外用户自己也可以开发排版功能（即自己书写宏包文件，参见第七章），在提交使用自定义的排版功能或环境创建的文档时，只要连同自己的宏包文件一起提交，就不用担心同文档接受者之间环境的一致性问题了。如果您用过 C 语言并自己写过包含文件，肯定很容易理解这一点。

1.2 输入文件的基本组成和格式

1.2.1 LaTeX 排版的一般过程

LaTeX 由于使用了纯文本表示排版文档的标注语言方式，所以从输入文档开始到得到最终排版结果为止，要经过编辑、编译产生设备无关的 dvi 文件。对于 dvi 文件可以使用工具进行预览打印，也可以将其转换成其他格式的文档（如 PostScript 或 pdf 等）。而在网络上传输或发布时，只需要使用纯文本的输入文档（我们经常也称之为 LaTeX 源文档）即可，这一点同 HTML 之类的标注语言工作方式类似，Web 页面文件都是文本文件，而经过各种 WWW 浏览器（可以由不同厂商提供、运行在不同平台上）进行“编译、预览”，用户看到的是“排版”后的页面，在浏览器中还可以打印页面文档。因此使用 LaTeX 系统排版文档时，输入文件的建立将是最主要也是最重要的工作，1.2.2 和 1.2.3 节将简单介绍一下 LaTeX 输入文档的组成和结构情况，本书以后的绝大部分内容都是帮助我们建立输入文件，这些文件经过“浏览”后能够产生各种我们所期望的文档输出。

1. 建立文档的输入文件

用户可以使用自己熟悉的一种文本编辑软件，将要排版的文档内容和控制排版的格式保留字输入计算机并存为纯文本文件。然而 300 多条的格式保留字对于非专业排版人士来说实在难以掌握，国内外的软件开发人员也相继开发出一些图形界面的 LaTeX 格式文档编辑工具，如目前在国内使用比较多的有 WinTex95、PcTex3.3、WinEdt、TexShell、EditPlus 等几种，它们都能运行于 Windows95/98 上并自带文本编辑器，使用户不需要输入文本格式的保留字。本书第九章介绍了其中的一些软件。

2. 编译输入文件并预览打印

LaTeX 系统用 TeX 程序和 LaTeX 宏包处理文档的输入文本文件，产生下面三类输出文件：

（1） 设备独立文件(*.dvi)

这个文件包含能翻译成各种输出设备输出指令的命令，用户可以用专门的预览软件（如 MikTeX 中的 Yap）来看这个文件的排版输出结果或者在各种打印机上输出，也可以使用专门的转换程序（如 MikTeX 中的 dvips 和 dvipdfm）转换成 PostScript 的 ps 文档或 Adobe Acrobat 的 pdf 文档，然后使用相应的阅读软件阅读。MikTeX 中的 pdfTeX 还可以直接从 LaTeX 源文件生成 pdf 文件。

（2） 脚本日志文件(*.log)

包含输入文档的汇总信息、诊断信息以及文档中发现的错误。

（3） 辅助文件(*.aux)

这是 LaTeX 内部使用的文件，帮助系统进行排版工作（如分节等等）。

目前较为常用的 LaTeX 编译系统有 MikTex 和 emTeX 等，本书的第九章也作了简单的介绍。在中国及东亚地区由于语言的原因，除了利用 LaTeX 提供的东亚文字支持软件包 CJK 外，目前出现了 CCT emTeX 等处理汉字 TeX 文档的工具。另外有些环境（如 Scientific Notebook 和 EditPlus）将文档输入与 LaTeX 编译集成在一个环境中完成，或者提供各种插

件使文档的编译过程对用户透明。

1.2.2 源文档组成元素

任何 LaTeX 输入文件都是纯文本文件，根据对排版结果的影响，其内容（当然都是文本字符）可以分为注释（不影响结果但可以提高输入文件的可读性）、空白符（控制文档排版时的间隔情况，影响的结果根据上下文环境会有所不同）、LaTeX 命令（最重要的组成部分，可以完成诸如版面设置、模式切换、公式符号输出等各种功能）和文档文本（少量的纯文字内容，大量的文档内容都作为某个命令的参数形式出现）。下面逐个进行简单的介绍。

1. 注释

如果 LaTeX 处理输入文件时遇到%符号，将忽视此行%符号以后的文本文字。因此我们可以用%作到行尾的注释，注释内容将不出现在文档的打印结果中。下面是一个例子，其中右边是输入文件中的文本，左边是格式化输出结果（后文中例子均按照此方式给出，不再另行说明）：

This is an example.

```
This is an % stupid
% Better: instructive <----
example.
```

2. 空白符

LaTeX 把诸如空格、制表符等统一当作空白符对待，若干连续的空白符也只作为一个空白符处理。一行开始处的空白符常常被忽略，而单个换行符作为一个空白符对待。两行文本之间的空行被当作一段文字的结束标志，多行空行的作用与一个空行的作用相同。例如：

```
It does not matter whether you enter
one or several spaces after a word.
An empty line starts a new para-
graph.
```

```
It does not matter whether you
enter one or several     spaces
after a word.

An empty line starts a new
paragraph.
```

3. LaTeX 命令

所谓命令就是 LaTeX 格式保留字或各种符号的文本表示。命令是大小写敏感的，并且只能是下面两种格式里的一种：

(1) 以反斜线\开头、后跟一个只由字母组成的命令名称。命令名称到后面第一个空白符、数字或其他非字母符号处结束。

(2) 由一个反斜线\和一个特殊符号组成。

LaTeX 忽略命令后面的空白符，如果你想在命令后面输出空格的话，要么输入{}再输入空格，要么使用特殊的留空命令。{}将阻止命令继续吃掉它后面的空白符。

有些命令的名称后需要带用一对花括号{}括起来的参数；有些命令支持后带用一对方括号[]括起来的可选参数。例如：

You can *lean* on me!
Please, start a new line right here!
Thank you!

```
You can \textsl{lean} on me!
Please, start a new line right
here!\linebreak[3] Thank you!
```

4. 文档的文本内容

(1) 普通字符的输入

普通字符是指文档预定输出内容中在标准 ASCII 码表中存在的那些字符（除了下面指出的几个特殊字符外），这些字符可以直接用键盘原样输入。

(2) 特殊字符的输入

下面这几个字符是 LaTeX 内部保留字符，有其特定的含义，不能直接在文档文本中输入它们，否则它们打印不出你所期望的结果。

$ & % # _ { } ~ ^ \

不过你可以用一个反斜线作为前缀来输入它们原来所代表的字符，如：

$ & % # _ { } \$ \& \% \# _ \{ \}

其他三个特殊字符前面加上反斜线后是命令名称，因此它们的输入要特殊处理，例如输入$\backslash$代表输入一个反斜线符号。要输入~或^字符可以使用字面打印环境或命令（参见 2.2.4 节）。另外如果你知道一个字符的 ASCII 编码，也可以使用\symbol 命令来表示字符：

\symbol{*char-code*}

其中参数 char-code 表示字符编码，可以使用十进制数，也可以使用十六进制数（此时 char-code 前面加双引号"）或八进制数（此时 char-code 前面加单引号'）。如空格的 ASCII 编码是 32，用十六进制表示为 20，用八进制表示是 40，因此\symbol(32), \symbol("20)和\symbol('40)都代表输入一个空格。

1.2.3 源文档组织结构

每一种标注语言的源文件都像高级语言程序一样，在开头部分说明本文件的目的、格式、与别的文档之间的关系（主要是打开必须使用的先导文档）等等，LaTeX 输入文档也不例外，也必须符合一定的结构。每个输入文件必须以命令\documentclass{...}开始，这条命令指明要书写的文档的类别。然后文档可以包含影响全局风格的命令，还可以装入包(packages)以便向 LaTeX 环境添加新的特性或功能。包的装入要使用命令\usepackage{...}。引导工作完成后，文档体部分以命令\begin{document}开始，现在可以以文字与必要的 LaTeX 命令混合的方式输入文档。文档的结尾加上命令\end{document}，告诉 LaTeX 忽略此命令以后的所有内容。

```
\documentclass{article}
\begin{document}
  Small is beautiful.
\end{document}
```

图 1.1 最小的输入文件

```
\documentclass[a4paper,11pt]{article}
\usepackage{latexsym}
\author{H.~Partl}
\title{Minimalism}
\frenchspacing
\begin{document}
\maketitle
\tableofcontents
\section{Start}
    Well and here begins my lovely article.
\section{End}
\ldots{} and here it ends.
\end{document}
```

图 1.2 一篇杂志文章框架

图 1.1 给出了一个最小的输入文件的内容，图 1.2 所代表的文件要复杂一些。\begin{document}命令之前的输入内容我们称之为导言区(preamble)，而界于\begin{document}和\end{document}之间的区域我们称之为文档区。下面我们分别介绍这些区域的主要用途或内容。

1. 导言区

(1) 装入必要的类和包文件

为了便于用户定制或扩充排版系统，LaTeX 将许多功能分类并存放在不同的类或包文件中。如果文档中要使用某些类和包文件中的功能，有必要在文档的导言区中声明。包文件由下面的命令激活：

\usepackage[*options*]{*package*}

其中 package 是要激活的包的名称，options 是一组关键字列表，这些关键字激活包中特定的特性或功能。有些包随 LaTeX2e 基本配置发布（见表 1.1），别的包则独立提供。关于类和包文件的内容请参见第七章。

表 1.1 一些随 LaTeX 发布的包

Doc：允许文档中嵌入 LaTeX 源代码
exscale：提供可变比例的数学公式扩充字体
fontenc：指定 LaTeX 系统应使用的字体编码
ifthen：提供 if...then do...otherwise do...形式的命令
latexsym：用以访问 LaTeX 符号字体
makeidx：提供建立文档索引的命令
syntonly：对文档不作排版处理
inputenc：允许指定输入文件的编码方式，如 ASCII, ISO Latin-1, ISO Latin-2, 437/850 IBM 代码页, Apple Macintosh, Next, ANSI-Windows 或用户自定义方式

(2) 指定文档类别

LaTeX 首先要了解所处理的文档类别，这一点由\documentclass 命令声明：

\documentclass[options]{class}

其中 class 指明所要创建的文档类别，例如

\documentclass[11pt,twoside,a4paper]{article}

表明要排版的文档是一篇文章，基本字符尺寸是 11 点(pt)，A4 纸双面打印。详细内容参见 2.1.1 节。

(3) 确定页面风格

\documentclass 命令决定了页眉页脚的大小和位置，页面风格命令则决定它们里面有些什么。详细内容参见 2.1.2 节。

(4) 进行文档全局性设置

有些设置将作为“标准”设置影响整个文档区的排版结果，如缺省的文本字体字号或色彩等等，如果你想使用特定的设置作为文档的缺省设置，可以将设置命令放在导言区行将结束的地方。

2. 文档区

(1) 处理模式

LaTeX 在处理输入文档文件时，总是处于下面三种模式中的一种：

- Paragraph（段落模式）
- Math（数学模式）
- Left-to-right（从左到右模式，简称 LR 模式）

段落模式是最常见的模式，LaTeX 处理普通文本文档时就是处于这种模式。段落模式下 LaTeX 将文档分成许多行，若干行文本又组成一页。当 LaTeX 生成数学公式时就处于数学模式。同处于段落模式时一样，处于 LR 模式的 LaTeX 也把处理的输出看作是许多单词组成的字符串，单词之间存在若干大小不等的空白。但是同段落模式不同的是，LR 模式下输出文本保持从左到右的顺序，从来不会另起一行。例如，即使你把成百上千个单词放进\mbox 所创建的一个文本盒子(box)中，LaTeX 也会继续试图将它们从左到右排到一个盒子里，并“抱怨”盒子太宽以至于一行中根本放不下。

一旦文档中用\mbox 命令创建一个盒子时，LaTeX 就开始进入 LR 模式。在盒子中也可以让系统进入另一种不同的模式，例如可以在盒子中放一个数学公式使之进入数学模式。还有许多文本生成盒子的命令和环境使 LaTeX 处于段落模式，此时生成的盒子称作 parbox。生成盒子时所处的段落模式我们称之为内部(inner)段落模式，而通常情况下的段落模式我们称之为外部(outer)段落模式。

只有当 LaTeX 在不同的处理等级间切换的时候才可能发生模式变化，尽管不是所有的等级切换都会引起模式变化。更具体地说，模式变化之可能发生在进入或离开一个环境的时候，或者 LaTeX 正在处理特定的文本产生命令的参数的时候。

(2) 大文档项目

当处理很大的文档文件时，有时我们希望将输入文件分割成若干部分，LaTeX 为此提供了两条命令。第一条命令是：

\include{*filename*}

在文档区我们可以使用这条命令插入另一个名叫 filename 文件的内容，LaTeX 在处理插入的文本以前将启动新的一个页面。

第二条命令可以在文档的导言部分中使用，允许LaTeX对可插入的文本文件进行限制。这条命令的格式是

\includeonly{*filename1*, *filename2*,...}

注意文件名 filename1,filename2,...同其后面的逗号之间不能有空格。这条命令执行后，上面的 \include{filename} 命令中的 filename 参数必须是 \includeonly{filename1, filename2, ...}命令参数中的一个。

\include 命令将在新的页面上对插入的文件文本进行排版，如果你不希望这样，可以用下面的命令插入别的文件：

\input{*filename*}

1.3 LaTeX 排版快速导游

本节我们使用 MiKTeX 带领读者完成一个简单的排版例子，目的是让大家对 LaTeX 排版过程有一个直观的印象。

1. 安装 MiKTeX 系统

按照 1.1.2 节给出的国内网址或 9.1.1 节末尾给出的国外网址，下载 MiKTeX1.20e（zip 压缩包的大小约为 22MB），然后解压到硬盘（假设为 C:\MiKTeX1.20）。在该目录下找到 setupwiz.exe 文件，运行该安装程序。

(1) 在第一个欢迎窗口中按下“下一步”按钮，将会要求你给出安装路径。假设我们输入 c:\texmf 作为安装 MiKTeX 的根目录（有关 MiKTeX 目录结构及配置详见 9.1 节）。

(2) 再按“下一步”后将进入安装组件选择窗口，作为初学者假设我们全部选中（缺省），所以可以直接往下进行。完全安装大约需要 50MB 的硬盘空间。

(3) 下一步安装向导将产生程序图标，可以使用其默认值并继续。

(4) 此时需要指定本地 TEXMF 根目录，详情见第九章。我们选择上面那个单选按钮并接受指定的 c:\localtexmf 目录名称。

(5) 在“附加 TEXMF 目录树(Additional TEXMF Directory Trees)”对话框中选中上面的选项（即无附加目录树）。

(6) 确认安装设置信息后，安装向导将完成 MiKTeX 的安装工作。此时在 Windows 的开始/程序/MiKTeX 程序组中应当有 DVI Viewer 图标（Yap 工具的快捷方式）。

安装完毕后还需要手工设置系统路径，在系统的 autoexec.bat 文件中加上下面一行：

```
set PATH=%PATH%;c:\texmf\miktex\bin
```

然后重新启动操作系统使你作的修改有效。

2. 输入并编译 LaTeX 源文件

使用任何你所熟悉的文本编辑软件，输入如下的文本内容(该例子在图 1.2 所示的 LaTeX 输入文件基础上略作改动)，并将它保存为 c:\texmf\tex\latex 目录下的 try.tex 纯文本文件：

```
\documentclass[a4paper,11pt]{article}
\usepackage{latexsym}
\author{Dayong Sang \and Ying Wang}
\title{A Sample Article}
```

```
\frenchspacing
\begin{document}
    \maketitle
    \section{Start}
    Our sample article contains an equation as following:
    \begin{displaymath}
        \mathop{\mathrm{corr}}(X,Y)=
        \frac{\displaystyle
        \sum_{i=1}^n(x_i-\overline x)
        (y_i-\overline y)}
        {\displaystyle\biggl[
        \sum_{i=1}^n(x_i-\overline x)^2
        \sum_{i=1}^n(y_i-\overline y)^2
        \biggr]^{1/2}}
    \end{displaymath}
    \section{End}
    \ldots{} and here it ends. It's easy, isn't it?
\end{document}
```

然后打开 Windows 的 DOS 命令窗口，进入 c:\texmf\miktex\bin（也就是源文件所在目录），运行下面的 DOS 命令：

```
latex   try
```

你会发现 latex 编译程序生成了几个以 try 为主文件名的新文件，目前我们最关心的是所产生的 try.dvi 文件。

3. 预览或打印 DVI 文件

运行前面我们提到过的 DVI Viewer（即 Yap 工具），打开 try.dvi 文件，放大到合适的倍数，我们将会看到如图 1.3 所示的文件内容。这就是 try.tex 的 LaTeX 排版结果。当然我们也可以使用 Yap 提供的打印功能将排版结果打印出来。

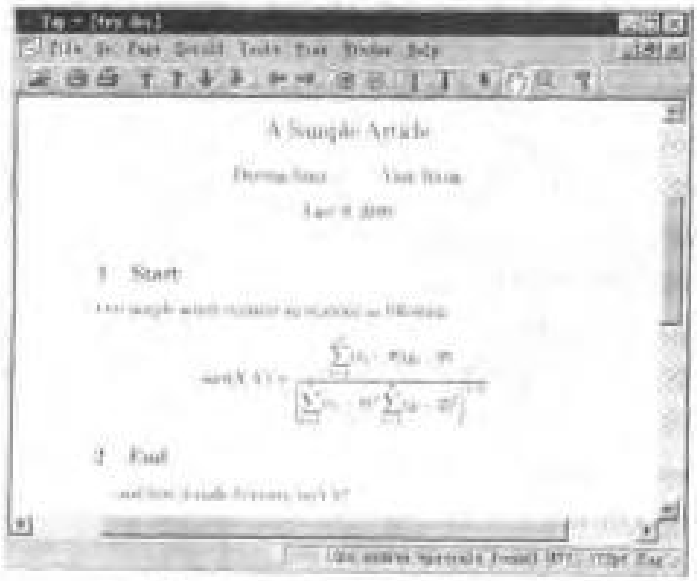


图 1.3 try.dvi 文件的预览效果

4. 转换成其他格式的文件

使用 MiKTeX 提供的 dvips 工具可以从 try.dvi 生成 PostScript 文件。在 DOS 命令窗口中执行下面的命令：

```
dvips  try
```

将产生一个新的文件 try.ps，读者可以用 GSView 打开该文件。另外使用 MiKTeX 中的 dvipdfm 工具可以从 try.dvi 生成 pdf 文件：

```
dvipdfm  try
```

读者同样可以使用 Adobe Acrobat Reader 打开所生成的 try.pdf 文件。图 1.4 和图 1.5 分别展示了 try.ps 和 try.pdf 的阅读效果。

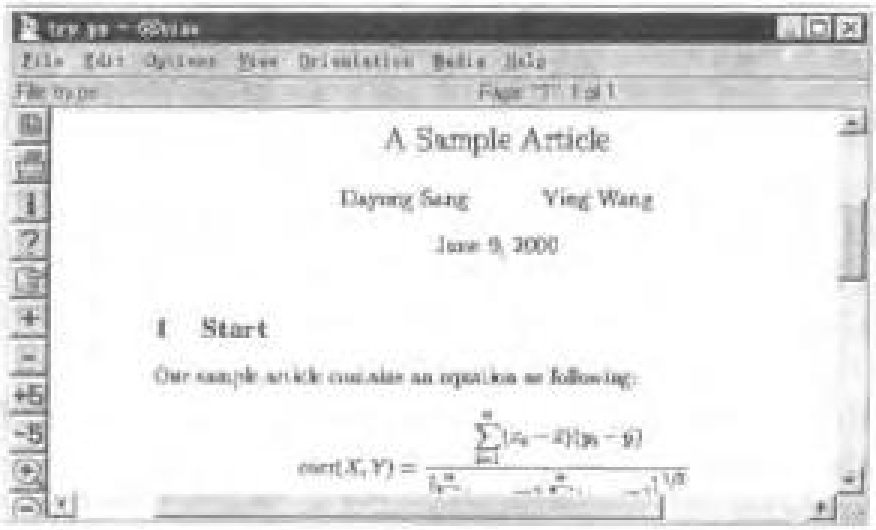


图 1.4 打开后的 try.ps 文件

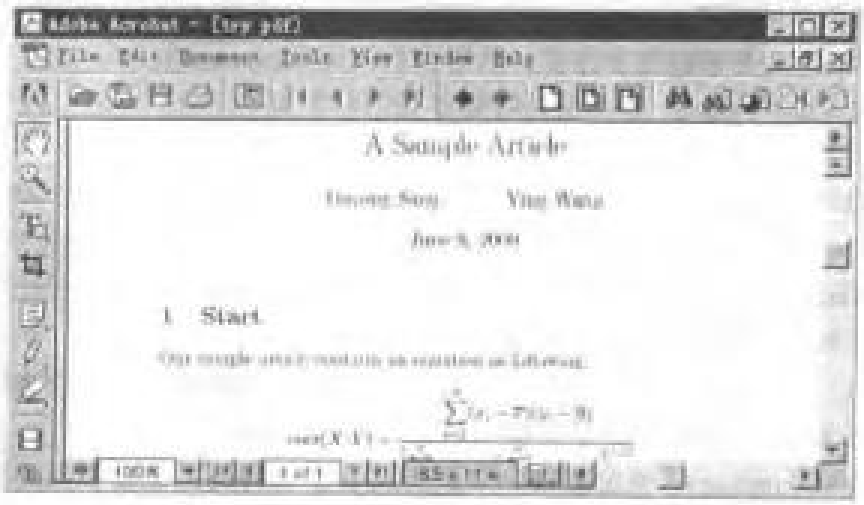


图 1.5 打开后的 try.pdf 文件

MiKTeX 还提供了一个工具 pdfTeX，该工具能够直接从 TeX 输入文件生成 pdf 文件，读者也可以试一下，但是由于它只能理解 TeX 而不是 LaTeX，对 try.tex 的处理结果肯定让你失望了。

第二章 文本文档的排版

阅读完前面一章以后，我们对 LaTeX 文档的基本情况已经有所了解了。为了书写实用的文档，本章将详细介绍 LaTeX 中输入文本文档时必须要用到的许多命令等。

2.1 文档类别和页面风格

2.1.1 文档类别

合法的 LaTeX 文档类别主要包括下面 5 种：

- article（文章）
- report（报告）
- letter（信函）
- book（书籍）
- slides（幻灯片）

除了幻灯片类别以外的标准文档类别都接受下面这种形式的可选参数确定字体大小（缺省大小为 10pt，1pt=1/72.27 英寸，关于长度单位参见 2.5.1 节）。

10pt, 11pt, 12pt

所有文档类别都接受下面这种形式的可选参数确定纸张大小（缺省为 letter）。

a4paper, a5paper, b5paper, letterpaper, legalpaper, executivepaper

还有一些杂类可选参数：

- landscape: 选择横向排版方式，缺省为纵向排版(portrait)。
- titlepage, notitlepage: 选择是否需要单独的标题页。
- leqno: 公式的编号放在公式的左边，缺省时放在右边。
- eqn: 公式显示左边对齐，缺省时中间对齐。
- openbib: 使用"open"方式的参考文献格式。
- draft, final: 标注/不标注带标尺的边框，缺省值为 final。

下面的可选参数不能用于幻灯片类别：

- oneside, twoside: 选择单面打印还是双面打印。书籍类别文档的缺省值为双面打印(twoside)，其余类别的缺省值为单面打印(oneside)。
- openright, openany: 确定新的一章是否从右手页开始，书籍类别的缺省值是从右手页开始(openright)。
- onecolumn, twocolumn: 排通栏还是排分栏（2 栏），缺省值为通栏(onecolumn)。

幻灯片类别的文档可以使用 clock 可选参数，使得每一页的底部打印当前时间。如果使用的可选参数不止一个，必须用逗号把它们分隔开来。

2.1.2　页面风格

页面风格决定了文档页中页眉页脚(header&footer)的格式和内容。下面我们依次介绍一些关于页面风格的命令。

1. \pagestyle 命令

\pagestyle{*style*}

LaTeX 支持三种预定义的页眉/页脚组合，每种组合称为一种页面风格。\pagestyle{*style*}命令中的 style 参数指明从当前页开始以后使用哪种页面风格，预定义的 style 值见表 2.1。

表 2.1　　LaTeX 预定义的页面风格

plain：缺省的页面风格，将页号打印在页面底部页脚的中央
headings：　将当前章次的标题及页号打印在每一页的页眉上，页脚保持空白
empty：页眉页脚均保持空白，不带页号
myheadings：需要你再使用\markboth 或\markright 命令指明页眉中的内容

2. \thispagestyle 命令

\thispagestyle{*style*}

这条命令仅仅改变当前文档页的页面风格而不影响以后的页面，参数style的含义同\pagestyle命令中的style参数。

3. \maketitle 命令

\maketitle

这条命令将除了文章类别以外的文档的标题产生到单独的一页中，文章的标题通常放在首页的顶部。

4. \author 命令

\author{*names*}

这条命令给出文档的作者姓名 names，如果不止一位作者，names 中各位作者的姓名之间用\and 命令隔开。一个作者项内部可以用\\分行，表示作者的供职机构和地址等信息。

5. \date 命令

\date{*text*}

这条命令将文档的日期设置成 text 参数所表示的日期。如果文档中没有使用此命令，LaTeX 将把当前日期作为文档的日期。

6. \thanks 命令

\thanks{*text*}

这条命令为文档标题产生一个脚注。

7. \title 命令

\title{*text*}

这条命令设置文档标题为字符串 text，如果标题很长，可以带\\命令进行换行。

8. \pagenumbering 命令

\pagenumbering{*num_style*}

这条命令指定页码的格式，num_style 参数代表页码格式，可取的值有：

- arabic: 阿拉伯数字
- roman: 小写罗马数字
- Roman: 大写罗马数字
- alph: 小写英文字母
- Alph: 大写英文字母

9. \markright 命令和\markboth 命令

\markright{*right head*}

\markboth{*left head*}{*right head*}

这两条命令同 myheadings 页面风格相配合，前者只设置右页页眉而让左页页眉保持原样，后者设置左右两个页眉。要注意：左页页眉由当前页结束前最后一条 markboth 命令决定；右页页眉则由当前页中第一个出现的\markboth 或\markright 命令决定，如果当前页没有这两条命令，则由此前页中最后一个\markboth 或\markright 命令决定。

2.2 LaTeX2e 命令

有关 LaTeX 的两种命令名称格式，我们在 1.3.3 节中已经作了简单的介绍，本节中我们将分类介绍 LaTeX2e 的各种命令。注意：LaTeX 命令是区分大小写的，如果输入时不注意大小写，将得到不可预知的结果。

2.2.1 换行换页与段落命令

文档在排版时往往要求每一行具有相同的长度，LaTeX 为了对整段的文档进行优化，将插入必要的换行和空格。如果必要的话对于一行中不好放的单词，LaTeX 还将用连字符把它分开放在两行之中。一般情况下每个文档段的首行按锯齿状缩进，段与段之间没有多余的空格。

1. 换行命令

(1) \\和*命令

\\ [*extra-space*]

* [*extra-space*]

有时出于某种需要我们有必要显式地插入换行，我们可以使用这两条命令。此时 LaTeX 将换至新的一行而不开始一个新的段落。*同\\命令功能相同，差别只是后者在强制换行后禁止换页。可选参数 extra-space 指明下一行开始前预留多少垂直距离（可以为负值）。

(2) \newline 命令

\newline

\newline 命令只能在段落中使用，使文本开始一个新行。

(3) \linebreak命令

\linebreak[*number*]

这条命令让 LaTeX 中断当前行并将当前行已有文本拉长直至页边。如果使用了 number

可选参数，则这条命令就变成了一个换行请求，换不换行由系统决定。number 的值只能从 0 到 4，当然值越大代表换行的意愿越强烈。

(4) \nolinebreak 命令

\nolinebreak[*number*]

这条命令与上面的\linebreak 命令正好相反，避免文档在命令处换行。可选参数 number 的含义同上一条命令里的 number 参数。

(5) 连字与断字命令

需要连字符的时候一般 LaTeX 都能够自动处理。如果 LaTeX 的断字算法找不到合适的连字点，我们可以使用下面的命令告诉系统怎样处理这种意外情况。

\hyphenation{*word list* }

word list 中列出需要单独考虑的单词列表，LaTeX 遇到需要对该列表中的某个单词进行连字符分割时，只有我们使用\-命令的地方允许断开。如果我们在一个单词的中间使用了\-命令，则 LaTeX 不再用断字算法为该单词寻找别的可行断点。列出的这些单词中不能含有特殊字符或符号，字母不论大小写处理方法都一样。下面的例子使系统按照指定的可行位置对"hyphenation"单词断字，同时禁止将"FORTRAN", "Fortran"或"fortran"断开：

\hyphenation{FORTRAN Hy-phen-a-tion}

多个单词要一起保持在同一行中，我们可以使用命令

\mbox{*text*}

这条命令使 text 参数中的多个单词组成的字符串在任何情况下都不会分开在两行中，如

```
My phone number will change soon.
It will be 0116 291 2319.
The parameter filename should con-
tain the name of the file.
```

```
My phone number will change soon.
    It will be \mbox{0116 291 2319}.
The parameter
\mbox{\emph{filename}} should
contain the name of the file.
```

(6) \sloppy 命令与\fussy 命令

\sloppy

\fussy

缺省时 LaTeX 采用\fussy 方式，这种方式的排版思想是尽量减少一行中单词之间的空格数量，但随之而来的缺点是断字情况较多。文档中可以使用\sloppy 命令是系统从此开始尽量减少断字情况，不过单词间空格可能会多起来。用户也可以用\fussy 命令结束前面出现的\sloppy 命令的作用，恢复\fussy 方式。

2. 换页命令

(1) \newpage 命令

\newpage

\newpage 命令结束当前页，开始新的一页。

(2) \pagebreak 命令

\pagebreak[*number*]

这条命令让 LaTeX 从文本当前位置结束当前页。如果给出了可选参数 number，则此

命令变成了一个请求，如何处理由系统决定。munber 可以取 0~4 的值，值越大表示意愿越强烈.

(3) \nopagebreak 命令

\nopagebreak[*number*]

这条命令同上一条命令的作用正好相反，禁止从当前位置结束当前页。number 参数的含义和取值同上一条命令的 number 参数相似。

(4) \enlargethispage命令

\enlargethispage{*size*}

\enlargethispage*{*size*}

这条命令按照指定的尺寸增大当前页面的\textheight参数，有关此参数的含义见3.3节。如下面的命令将使当前页面的高度增大额外的一行。

\enlargethispage{\baselineskip}

星号版的命令尽量将当前页面中的内容往一块儿压缩，通常与一条显式的换页命令\pagebreak 一起使用。

3. 段落命令

所谓段落(paragraph)就是用一个完全空白的行（甚至连一个注释标志%都没有的空行）结束的一段文本。空行不能出现在不允许段落开始的时机或地方，如数学状态下或分节命令的参数中等等。

(1) \indent 命令

\indent

这条命令产生一块水平空白区域，其宽度等于段落的缩进距离值。在段落缩进被忽略或禁止的地方，如果想得到缩进的效果的话，可以使用这条命令。

(2) \noindent 命令

\noindent

此命令如果出现在段落的开头，将禁止段落的缩进；如果出现在段落的中间则不起作用

(3) \par 命令

\par

这条命令与一个空行的效果相同，使用它常常只是为了使文档中的命令或环境的定义更易于阅读理解。

2.2.2 计数器命令

有许多命令和环境会导致 LaTeX 进行编号，对于每一种编号，LaTeX 都会用一个计数器记录编号情况。计数器的名称同导致编号的命令或环境相同，二者的差别就是计数器的名称不带反斜线，不过 enumi-enumiv 是个例外，它们用来给嵌套的 enumerate 环境计数。下面列出的是 LaTeX 标准文档类别中控制编号的计数器名称。

part	paragraph	subparagraph	figure	equation	enumi
chapter	section	subsection	subsubsection	table	enumii
page	footnote	enumiii	mpfootnote	enumiv	

下面我们介绍一些有关计数器的命令。

1. \addtocounter 命令

\addtocounter{*counter*}{*value*}

这条命令将计数器 counter 的值增加 value 大小，value 参数的值可以为负数。

2. \alph 命令和\Alph 命令

\alph{*counter*}

\Alph{*counter*}

这两条命令使得计数器 counter 的值以英文字母的方式打印。前者用小写字母编号，即 a, b, c...；后者用大写字母编号，即 A, B, C...

3. \arabic 命令

\arabic{*counter*}

这条命令使得计数器 counter 的值以阿拉伯数字的方式打印，即 1, 2, 3...

4. \fnsymbol 命令

\fnsymbol{*counter*}

这条命令用下面 9 种脚注符号来表示脚注计数器 counter，counter 的值只能取 1~9，依次对应每一个符号：

* † ‡ § ¶ • ** †† ‡‡

5. \newcounter 命令

\newcounter{*foo*}[*counter*]

这条命令定义一个新的计数器名叫 foo 并将其值初始化为 0。可选的参数 counter 为另一个计数器，每当 counter 的值增加时，foo 计数器就将被复位。

6. \roman 命令和\Roman 命令

\roman{*counter*}

\Roman{*counter*}

这两条命令使计数器 counter 的值以罗马数字的方式打印。前者使用小写罗马数字，即 i, ii, iii...后者使用大写罗马数字，即 I, II, III...

7. \stepcounter 命令和\refstepcounter 命令

\stepcounter{*counter*}

\refstepcounter{*counter*}

\stepcounter 命令将计数器 counter 的值加 1，同时复位所有的辅助计数器。\refstepcounter 命令除了完成\stepcounter 命令的功能以外，还将\ref 的当前值作为\thecounter 的结果。

8. \setcounter 命令

\setcounter{*counter*}{*value*}

这条命令将计数器 counter 的值设置为 value。

9. \usecounter 命令

\usecounter{*counter*}

这条命令用于 list 环境的第二个参数，指明用 counter 作为 list 列表项的计数器。

10. \value 命令

\value{*counter*}

这条命令获取计数器 counter 的当前值（整数编号），有时候对计数器进行算术运算是很有用的。下面是一个例子：

\hspace{\value{foo}\parindent}

2.2.3 交叉引用命令

我们对图、公式等进行编号的目的之一，就是为了让读者能够按照编号索引到它们，如我们的文档中可能会有这样的文字："详细情况参见图 3"，我们希望读者此时可以索引到图 3 所在的文档位置。

1. \label 命令

\label{*key*}

这条命令定义标号。原始文本中的\label 命令将当前文档单元的编号赋给标号 key；已编号的环境中的\label 命令将环境编号赋给标号 key。key 可包括字母、数字和标点符号的任意组合且大小写字母敏感。为了避免两个标号重名，通常使用"前缀：后缀"的形式对标号命名，习惯上我们使用的前缀有下面这些，如一个图的标号可以叫做 fig:bandersnatch。

- cha 代表章次
- sec 用于更低层的分节
- fig 用于代表图的标号
- tab 用于代表表的标号
- eq 用于代表公式的标号

2. \pageref 命令

\pageref{*key*}

这条命令得到标号 key 相对应的\label(key)语句所在处的页号。

3. \ref 命令

\ref{*key*}

这条命令得到 key 所代表的分节单元、公式等等的编号。下面是一个例子：

```
A reference to this subsection looks
like: "see section 2.6 on page 20."
```

```
A reference to this subsection
    \label{sec:this} looks like:
``see section~\ref{sec:this} on
page~\pageref{sec:this}."
```

2.3 LaTeX2e 环境

环境的作用是定义一个文档区域，位于其内的文档可以根据环境参数进行不同的处理，譬如有可能暂时改变某一处理特征（如缩进，行宽，字样等等），这些改变只在该环境内有作用。环境的定义如下：

```
\begin{environment}
    text
\end{environment}
```

其中 environment 是环境的名称。环境可以嵌套若干次，只要文本中能够维持正确的开始和结束的顺序，如

\begin{*aaa*}...\begin{*bbb*}...\end{*bbb*}...\end{*aaa*}

下面的各节中我们逐次介绍一些环境，有些复杂的环境我们将在本章的后面专门介绍。

2.3.1　flushleft，flushright 和 center 环境

flushleft 用于产生左对齐的文档，flushright 用于产生右对齐的文档，center 环境用于产生中间对齐（居中）的文档。如果输入文件中没有用\\明确指定换行位置，LaTeX 将根据页面情况自动确定换行。下面是一些例子。

This text is
left aligned. LaTeX is not trying to
make each line the same length.

This text is right
aligned. LaTeX is not trying to make
each line the same length.

At the center
of the earth

```
\begin{flushleft}  This text is\\ left aligned.
\LaTeX{} is not trying to make
each line the same length. \end{flushleft}
\begin{flushright} This text is right\\ aligned.
\LaTeX{} is not trying to make
each line the same length. \end{flushright}
\begin{center} At the center\\of the earth
\end{center}
```

2.3.2　quote，quotation 和 verse 环境

quote 环境用于引文、重要的句子以及示例等，如：

A typographical rule of thumb for the
line length is:

> No line should contain
> more than 66 characters.

This is why LaTeX pages
have such large borders
by default.

That's why multicolumn print is often used in newspapers.

```
A typographical rule of thumb
for the line length is: \begin{quote}
No line should contain more than
66 characters.

This is why \LaTeX pages have
such large borders by default. \end{quote}

That's why multicolumn print is
often used in newspapers.
```

verse 环境同 quotation 环境类似。quotation 用于跨越若干段落的长引文，因为它能够进行段落缩排；而 verse 环境适合于诗歌体裁的排版，因为这时换行极其重要，每一行之间用行尾的\\符号分隔，每个 verse 环境后留一空行。见下例：

I know only one English poem by
heart. It is about Humpty Dumpty.

Humpty Dumpty sat on
a wall:
Humpty Dumpty had a
great fall.

```
I know only one English poem by
heart. It is about Humpty Dumpty.
\begin{flushleft}
\begin{verse}
Humpty Dumpty sat on a wall:\\
Humpty Dumpty had a great fall.\\
All the King's horses and all
```

```
All the King's horses
 and all the King's
 men
Couldn't put Humpty
 together again.
```

```
the King's men\\
Couldn't put Humpty together
again.
\end{verse}
\end{flushleft}
```

2.3.3 字面打印(verbatim)环境

位于\begin{verbatim}和\end{verbatim}之间的文本就像从打字机里打出来一样将被直接打印输出，不管是换行还是空白符都原封不动地照单打印，LaTeX 也不执行任何命令。

在一文档段中可以用下面的命令达到相似的效果：

\verb + *text* +

其中的 text 部分将直接输出。+只是我们随便找的一个分界符例子，其实除了字母、星号(*)和空白以外的所有字符都可以用作分界符。例如

```
The \ldots command . . .
```

```
The \verb|\ldots| command \ldots
```

```
10 PRINT "HELLO WORLD ";
20 GOTO 10
```

```
\begin{verbatim}10 PRINT "HELLO WORLD ";
20 GOTO 10 \end{verbatim}
```

```
the starred version of
the        verbatim
environment emphasises
the spaces     in the text
```

```
\begin{verbatim*} the starred version of
the        verbatim
environment emphasises
the spaces     in the text \end{verbatim*}
```

\verb 命令还可以带一个星号，效果相似：

```
    like     this :-)
```

```
\verb*|like     this :-) |
```

verbatim 环境和\verb 命令不能在别的命令的参数中使用。

2.3.4 微型页面(minipage)环境

```
\begin{minipage}[position]{width}
    text
\end{minipage}
```

此环境同 3.3 节中将要介绍的\parbox 命令功能相似，它们有着相同的可选参数和必选参数 width，参数含义参见\parbox 命令。同\parbox 命令最大的不同是在此环境中可以使用其他创建段落的环境。

minipage 环境中的脚注特别适合于放置图形或表格中的脚注，此时用\footnote 或\footnotetext 等脚注命令产生的脚注不是放在页面的页脚位置，而是放在 minipage 环境的底部位置，另外脚注计数器也不用通常的那个脚注计数器而用专门的 mpfootnote 计数器。注意：如果你在 minipage 环境中使用了脚注，则不要将该环境放入另一个 minipage 环境内部，否则脚注可能会放不到正确的 minipage 环境底部。

2.3.5 制表位(tabbing)环境

```
\begin{tabbing}
```

text

\end{tabbing}

使用这种环境可以在文本 text 中任一行文字中的任何位置设置 tab 停止点(tab stop)或称制表位，后续行中的文本就可以使用某些命令控制它们在这些停留点处显示。开始时我们可以认为制表位环境中只有一个制表位即文本区域的左边界。

1. 制表位的设置命令

(1) \= 命令

这个命令在行中当前位置设置一个制表位。设置的制表位按\=命令出现的先后排序。由于在制表位环境外部\=命令表示注音符号，所以如果制表位环境内要使用这类注音符号的话必须使用替代命令\a=。

(2) \> 命令

这个命令使下一个文本字符的起始输出位置跳至后面第一个制表位处。例如：

```
\begin{tabbing}
MON\qquad\= TUE\qquad\= WED\qquad\= THU\qquad\= FRI\\
Math    \> English \> Math \> Physics \>English\\
Computer \> Physics \> Music \> Computer \> Sports\\
\end{tabbing}
```

将产生如下的输出结果：

MON	TUE	WED	THU	FRI
Math	English	Math	Physics	English
Computer	Physics	Music	Computer	Sports

注意\>命令后移到下一个逻辑制表位位置，假设前一制表位后面的内容宽度超出这两个制表位间的距离，即当前位置比逻辑上的下一个制表位更靠右，则输出位置将回移。

(3) \pushtabs 和\poptabs 命令

\pushtabs 命令将当前行中的制表位设置情况压入堆栈保存并清除当前行中的制表位设置；\poptabs 命令将堆栈栈顶处的制表位设置恢复并作为当前设置。\pushtabs 和\poptabs 命令可以多次使用但必须成对出现，另外\poptabs 命令恢复制表位设置时按照后入先出的顺序进行。

2. 行首定位命令

(1) \+ 命令

使后续的文本行行首右移一个制表位。

(2) \- 命令

使后续的文本行行首左移一个制表位，但最多只能移到制表位环境的左边界处。从制表位环境的左边界算起，行首实际跳过的制表位是整个环境中迄今为止\+命令总数减去\-命令总数的差。

(3) \< 命令

使当前行忽略前面出现的\+和\-命令，从制表位环境的左边界开始并拥有被\+所跳过的全部制表位。

3. 行中文本定位命令

(1) \' 命令

left-text \' *right-text*

将 right-text 文本从当前制表位处开始输出，而将 left-text 放在 right-text 文本的左边(其右边与 right-text 的距离为\tabbingsep)，即使 right-text 处于行首也一样。\tabbingsep 距离的大小可以使用\setlength 命令设置。由于在制表位环境外部\=命令表示注音符号，所以如果制表位环境内要使用这类注音符号的话必须使用替代命令\'。

(2) \` 命令

使当前行从这个命令出现的位置开始至行尾处的文字与右边界右对齐输出。要求右对齐的文本中不能再含有\>或\=命令。由于在制表位环境外部\=命令表示注音符号，所以如果制表位环境内要使用这类注音符号的话必须使用替代命令\a`。

4. 不作输出的样本行

细心的读者可能会发现，使用前面方法设置的制表位不能控制第一行文本的输出，因为第一行设置的制表位从第二行开始才起作用。为此 LaTeX 允许我们在第一行输出文本的前面增加一个“假行”(注意不要同空行混淆)，并在这一行中设置制表位。\kill 命令代替\\命令结束样本行，同时也是样本行与普通文本行的区别标志。例如：

```
\hspace*{3cm}\=sample line col1 \=\hspace{4cm}\= \kill
```

注意行首要预留空白的话必须使用的\hspace*命令而不能使用\hspace 命令 (如上例所示)，否则会去掉在行边界所插入的间距。

5. 其他注意事项和应用实例

(1) 注意事项

制表位环境虽然也是一种文本模式，但是同通常的段落模式相比有许多特殊性，使用时要注意以下几点：

- 系统自动确定换页，禁止用户使用\newpage 和\clearpage 命令，忽略\pagebreak 命令的功能。如果要想分页，可以使用较大数值的垂直间距强迫系统自动分页，这样前一页最后一行距页尾的距离虽然不定，但是后一页页首一定不留垂直空白。
- 除非使用\\换行命令，系统不会自动断行，所以用户要控制每一行中的文本长度，同时系统忽略\hfill, \hrulefill 和\dotfill 命令的功能。

(2)应用实例

```
Apples:      consumed by: people          \begin{tabbing}
                          horses          Grapefruit: \= \kill
                        and sheep         Apples: \> consumed by: \= people\+\+\\
             reasonably juicy             horses \\
Grapefruits: a delicacy                   and \' sheep\-\\
(see also:   melons                       reasonably juicy\-\\
             pumpkins)                    Grapefruits: \> a delicacy\\
Horses       feed on          apples      \pushtabs
                                          (see also: \= melons\\
                                          \> pumpkins)\\
```

```
\poptabs
Horses \> feed on \> apples
\end{tabbing}
```

2.4　自定义命令和环境

有时候LaTeX没有提供完成你所需要功能的命令或环境，为了达到你要求的排版效果，必须使用多条命令或环境进行组合。如果这个功能在文档中经常或多处出现，这种组合将以相同的方式反复重复。为了简化工作量同时扩充排版能力，LaTeX 允许用户向语言中添加自己定义的命令或环境。

1. \newcommand 命令

\newcommand{*cmd*}[*args*]{*definition*}
\newcommand{*cmd*}[*args*][*default*]{*definition*}
\renewcommand{*cmd*}[*args*]{*definition*}
\renewcommand{*cmd*}[*args*][*default*]{*definition*}

这些命令定义一条新的命令或者重新定义一条命令，其中各个参数的含义如下：

cmd: 以反斜线开头的命令名称。对于\newcommand 命令来说 cmd 不能是已经有定义的命令名称同时也可以不以反斜线开头；对于\renewcommand 命令来说 cmd 必须是已经有定义的命令。

args: 1 到 9 之间的一个整数，说明 cmd 命令所需的参数数目，缺省时命令不带参数。

default:如果带有此参数则 cmd 的第一个参数是可选参数，且其缺省值就是 default 的值。

definition: 每当 cmd 命令出现时用以替换的文本；#n 表示 cmd 命令的一个形式参数，替换时要用 cmd 命令中的第 n 个实际参数的文本进行替换。

2. \providecommand 命令

这条命令的格式和作用同 newcommand 命令，惟一的区别在于如果新定义的命令名称已经存在，LaTeX 将不出错，而只是忽略该条命令。

3. \newenvironment 命令

\newenvironment{*nam*}[*args*]{*begdef*}{*enddef*}
\newenvironment{*nam*}[*args*][*default*]{*begdef*}{*enddef*}
\renewenvironment{*nam*}[*args*]{*begdef*}{*enddef*}

这些命令定义一个新的环境命令或者重新定义一个现有环境命令，其中各个参数的含义如下：

nam: 环境名称。对于\newenvironment 命令来说 nam 不能是已经有定义的环境名称，\nam 也不能是已定义的命令名称；对于\renewenvironment 命令来说 nam 必须是已经有定义的环境。

args: 1 到 9 之间的一个整数，说明新定义的环境所需要的参数数目，缺省时不带参数。

default:如果带有此参数则环境的第一个参数是可选参数，且其缺省值就是 default 的值。

begdef: 每当\begin{nam}命令出现时用以替换的文本；#n 表示环境的一个形式参数，替换时要用环境命令中的第 n 个实际参数的文本进行替换。

enddef: 每当\end{nam}命令出现时用以替换的文本，不能带任何参数。

4. \newtheorem 命令

\newtheorem{*env_name*}{*caption*}[*within*]

\newtheorem{*env_name*}[*numbered_like*]{*caption*}

这条命令最多只能带一个可选参数，用于定义一个模仿定理描述的环境，其中各个参数的含义如下：

env_name: 所定义的环境名称（字母串），不能是已有环境名或计数器名。

caption: 在环境开头、紧靠编号后面打印的文本，例如可以简单地只打印"Theorem"字样。

within: 一个已定义的计数器名称（通常属于分节类型），使得新的 theorem 计数器在 within 所代表的分节中能够复位。

mumbered_like: 一个已经定义的模仿定理描述的环境的名称。

5. \newfont 命令

\newfont{*cmd*}{*font_name*}

这条命令定义一个未被定义过的申明命令 cmd，cmd 将选择 font_name 参数所代表的字体作为当前字体。

2.5 长度与距离

2.5.1 固定长度

长度是由前面可能有符号（+或-）的小数，后接一个必需的尺寸单位组成。下面是可允许的单位及缩写名称：

cm 厘米，

mm 毫米，

in 英寸（1 in = 2.54 cm），

pt 点（1 in = 72.27 pt），

bp 大点（1 in = 72 bp），

pc pica（1 pc = 12 pt），

dd didôt 点（1157 dd = 1238 pt），

cc cicero（1 cc = 12 dd），

em 与字体相关的尺寸，相当于大写字母 M 的宽度，

ex 与字体相关的尺寸，相当于小写字母 x 的高度。

小数点可以用圆点句号，也可以用逗号，即 12.5cm 和 12,5cm 是一样的效果。注意 0 不是一个合法的长度，因为其没有长度单位。要表示零长度，必须用 0pt 或 0cm 等等。

2.5.2 橡皮长度

所谓橡皮长度，指这个长度可以伸展或收缩给定的量。橡皮长度的语法是：

normal-len plus *b-len* minus *s-len*

其中 normal-len, b-len 和 s-len 都是固定长度，表示该橡皮长度正常情况下为 normal-len，

但最大可以拉长到 normal len 加上 b-len 的和，最小可以收缩到 normal-len 减去 s-len 的差。例如：

\setlength{\parskip}{1ex plus0.5ex minus0.2ex}

的含义为将\parskip 设置为等于当前字体中 x 的高度，但是该行距可以伸长到 1.5 倍或收缩到 0.8 倍这个长度

下面是另一些使用橡皮长度概念的命令。

1. \dotfill 命令

这条命令产生一串橡皮长度的点而不是空格。

2. \hfill 命令

这条命令产生一个可水平方向伸缩的、用空白填充的橡皮长度。

3. \hrulefill 命令

这条命令产生一个可水平方向伸缩的、用水平标尺填充的橡皮长度。

4. \vfill 命令

这条命令产生一个可垂直方向伸缩的橡皮长度。

2.5.3. 长度命令和长度设置命令

1. 预定义的文本字符长度命令

\width

\height

\depth

\totalheight

这些命令可以作为测量文本字体尺寸的长度命令，其中\totalheight =\height +\depth。

2. 长度定义命令

\newlength{*gnat*}

这条命令将其必选参数\gnat 定义为长度命令并将其值设置为 0in。如果\gnat 命令已经存在的话，将会产生错误。

3. 长度设置命令

\setlength{*gnat*}{*length*}

这条命令用于将长度命令\gnat 的值设置为 length。

4. 长度增加命令

\addtolength{*gnat*}{*length*}

这条命令使得长度命令\gnat 的值增加一个增量 length，length 可以为负值。

5. 取文本深度命令

\settodepth{*gnat*}{*text*}

这条命令将长度命令\gnat 的值设置为同文本参数 text 的深度相等。

6. 取文本高度命令

\settoheight{*gnat*}{*text*}

这条命令将长度命令\gnat 的值设置为同文本参数 text 的高度相等。

7. 取文本宽度命令

\settowidth{*gnat*}{*text*}

这条命令将长度命令\gnat 的值设置为同文本参数 text 的宽度相等。

2.6 标题和章节

2.6.1 分章节

为了更好地组织文档，通常我们都将文档划分成若干章节以及层次更低的小节等等。LaTeX 提供专门的命令支持这种排版操作，每条命令都将章节的标题作为参数，用户的责任是按照正确的顺序使用这些命令。

如果文档类别是文章(article)，则可以使用下面这些分章节的命令：

\section{...} \paragraph{...}

\subsection{...} \subparagraph{...}

\subsubsection{...} \appendix

如果文档类别是报告(report)或书籍(book)，则除了上面这 6 条命令以外，还可以使用下面这两条命令：

\part{...} \chapter{...}

因为文章文档不识别 chapter 命令，所以很容易就可以将一篇文章作为一章内容插入到书籍文档之中。节与节之间的空格、节的编号以及标题的字体大小，都由 LaTeX 自动进行设置。

下面这两条分节命令有一些特别：

- \part 命令不影响各章次的编号顺序；
- \appendix 命令不需要参数，它只是将章次的编号改成用字母表示。

除了上面介绍过的分章节命令以外，LaTeX2e 为书籍类文档另外引入下面 3 条附加命令：

\frontmatter \mainmatter \backmatter

这些命令有助于分割你的作品，使得书籍中的章节题目和页码格式更能符合你的愿望。

2.6.2 建立目录表

通过扫描整个文档，LaTeX 可以创建由章节题目和相应页码构成的内容目录表，只要你在需要目录表的地方使用下面的命令就行了：

\tableofcontents

LaTeX 为目录表生成一个标题，但是目录表结束后并不自动换页。如果要换页可以在\tableofcontents 命令后面跟上一条\newpage 命令。

一个文档至少要被 LaTeX 扫描两遍才能生成正确的目录表，有时候可能还需要扫描第三遍（不过真有此必要的话系统会通知你的）。上一节中所有分章节命令都存在相应的星号版命令，所谓的星号版命令就是在命令名后跟上一个星号所构成（如\section{*Help*}的星号版命令就是\section* {*Help*}），这些星号版的分章节命令所产生的章节将不被编号，其题

目也不出现在目录表中。

通常情况下，目录表中显示的章节题目就是章节定义时所给定的章节题目，有时候章节题目过长而不适于在目录表中显示，定义章节时我们可以在真正的题目之前给出一个可选参数，这个参数的内容将代替章节题目出现在目录列表中。如

```
\chapter[Read it! It's Exciting]{This is a very long  and especially boring title}
```

有两条与\tableofcontents 命令相似的命令：\listoffigures 命令和\listoftables 命令。它们分别用来产生文档中所有图形和表格目录列表，工作方式与使用方法与\tableofcontents 命令完全相同。

注意：如果想使用产生目录的命令的话，文档中一定不能有\nofiles 命令。

2.7　脚注与边注

2.7.1　脚注

脚注出现在当前页的底部，可以用两种方式产生。一种用单个命令\footnote，另一种用\footnotemark 和\footnotetext 两条命令的组合。

1. \footnote 命令

\footnote[*number*]{ *text*}

这条命令将带编号的脚注文本 text 显示在当前页的底部，例如

Footnotes[α] are often used by people using LaTeX

```
Footnotes\footnote{This
is a footnote} are often used
by people using \LaTeX
```

[α] This is a footnote

可选参数 number 用来改变缺省的脚注编号。这个命令只能在一般文本段落中使用，不能在诸如\chapter 等分节命令或图形、报表环境中使用。

2. footnotemark 命令

这条命令在文本中输出脚注编号，而且可以在\footnote 命令不能使用的地方使用。脚注文本由\footnotetext 命令另行给出。这条命令可以产生代表同一个脚注的多个连续的脚注标记，方法是在第一个\footnote 命令之后使用命令

\footnotemark[\value{footnote}]

3. \footnotetext 命令

\footnotetext[*number*]{*text*}

这条命令指定当前页底部放置的文本内容，可以在\footnotemark 命令之后的任何时候使用，但是\footnote 语句不能使用的地方这条语句也不能使用。可选参数 number 的值用来替换缺省产生的脚注编号。

2.7.2　边注

下面的边注命令可以在页面的边注区域添加注解：

\marginpar[*left*]{*right*}

边注的第一行将与边注命令所在的文本行同高。如果你只给出必选参数 right，边注的文本将根据文档的不同风格进行不同处理：

- 单面打印风格的文档将边注内容放在右边的边注区域；
- 双面打印风格的文档将边注内容放在外侧的边注区域；
- 分栏排版的文档将边注内容放在较近一侧的边注区域。

通过使用\reversemarginpar 命令，可以强制边注文本放置到同上述原则确定的边注区域相对的另一个边注区中。如果两个参数都给出了，可选参数为 left 则使用左边边注区，为 right 则使用右边边注区。

通常边注的第一个单词不能带连字符，如果你想让系统放开这个限制，可以在注释文本的第一个单词前面加上\hspace{0pt}命令。

2.8 文本的强调与词间空格

2.8.1 文本的强调

传统方式下用打字机打印手稿时，重要的单词用带下划线来表示。LaTeX 文档中允许强调重要的单词或文本，此时使用下面的命令：

\emph{*text*}

text 即为要强调的文本，LaTeX 将用强调字体（通常为斜体）来显示这些被强调文本。如果被强调的文本中有些部分已经被强调过一次，则这些部分将用正常的正体字体显示以示强调　例如

If you use emphasising *in an already emphasised text, then* LaTeX *uses an* upright *font for emphasising.*

```
\emph{If you use \emph{emphasising} in an
 already emphasised text, then
\LaTeX{} uses an \emph{upright} font for
emphasising.}
```

2.8.2 词间空格

为了得到笔直的页面边界效果，LaTeX 在每行的各单词间插入了数目不等的空格。每一个句子结束后的位置插入的空格适当多一些，这样使得文档更容易阅读。LaTeX 认为句子结束的标志是句点符号、问号或惊叹号，如果句点跟在大写字母的后面则不认为是句子的末尾而认为是常用的缩写格式。如果用户认为这种假设有例外情况则必须明确指出：文本中一个空格前面若加上反斜线\，表示该处留一个空格且大小不能被放大；波浪线~表示此处产生一个不能放大的空格，且不能在此处换行；句点前面的\@命令表示尽管这个句点跟在大写字母的后面，但仍然是一个句子的结束符号。例如

Mr. Smith was happy to see her
cf. Fig. 5
I like BASIC. What about you?

```
Mr.~Smith was happy to see her\\
cf.~Fig.~5\\
I like BASIC\@. What about you?
```

用下面的命令可以让 LaTeX 处理句点后面的空格：

```
\frenchspacing
```

这条命令告诉 LaTeX 在句点后面不要放比一般字符后面更多的空格，许多非英语文档中这个命令很重要。如果使用了这个命令，\@命令就是不必要的了。

2.9　特殊字符和符号

2.9.1　双引号

我们不能直接输入"作为双引号使用，LaTeX 用两个连续的`表示左双引号，用两个连续的'表示右双引号。如

```
"Please press the 'x' key."          ``Please press the 'x' key."
```

2.9.2　破折号和连字符

LaTeX 懂得 4 种破折号，我们可以用不同数量的连续的减号来表示三种破折号，而第四种是数学中的减号（根本就不能算破折号）。下面是破折号使用的例子。

```
daughter-in-law, X-rated          daughter-in-law, X-rated\\
pages 13 – 67                     pages 13--67\\
yes—or no?                        yes---or no? \\
0, 1 and -1                       $0$, $1$ and $-1$
```

由上面的例子我们可以看出这 4 种破折号的输入表示：-表示连字符，--表示短破折号，---表示长破折号，$-$表示减号或负号。

2.9.3　省略号

在 LaTeX 中有些标点符号（如句点）只占很小的位置，因此如果我们用连续的三个句点来表示省略号的话，那么省略号的尺寸就不对了。LaTeX 提供了专门的省略号输入命令

```
\ldots
```

例如

```
Not like that ... but like that:      Not like that ... but like that:\\
New York, Tokyo, Budapest, . . .      New York, Tokyo, Budapest, \ldots
```

2.9.4　重音符号和其他特殊符号

LaTeX 支持许多种语言中使用的重音符号和其他特殊字符，如表 2.2 所示。

下面是一段示例文档：

```
Hôtel, naïve, élève,                  H\^otel, na\"\i ve, \'el\`eve,\\
smørrebrød, ¡Señorita!,               sm\o rrebr\o d, !`Se\~norita!,\\
Schönbrunner Schloß Straße            Sch\"onbrunner Schlo\ss{}
                                      Stra\ss{}e
```

表 2.2 LaTeX 中的重音符号、其他特殊字符及其输入文本

特殊字符	输入方式	特殊字符	输入方式	特殊字符	输入方式	特殊字符	输入方式
ò	\`o	ó	\'o	ô	\^o	õ	\~o
ō	\=o	ȯ	\.o	ö	\"o		
o̯	\u o	ȅ	\v c	o͡o	\H o	ç	\c c
o̲	\d o	o̲	\b o	o͞o	\t oo		
œ	\oe	Œ	\OE	æ	\ae	Æ	\AE
å	\aa	Å	\AA				
ø	\o	Ø	\O	ł	\l	£	\L
ı	\i	£	\j	¡	!`	¿	?`

2.10 列表环境

2.10.1 列表环境分类

根据列表项中缺省标识的不同，列表环境可以分为三种，其环境名也不同。每种环境中的列表项至少要有两项。

1. 黑点列表环境

```
\begin{itemize}
    \item [label] item_1
    .....
    \item [label] item_n
\end{itemize}
```

这个环境建立由 n 项列表项组成的列表，其中每个列表项以一个黑圆点作标识，列表项内容用\item 命令的参数一个一个指定。可选参数 label 可以强制指定标识符号，详见本小节后面的内容，这里我们假设不用它。例如：

- The first item of an itemize
- The second item of an itemize

```
\begin{itemize}
    \item The first item of an itemize
        \item The second item of an itemize
\end{itemize}
```

2. 编号列表环境

```
\begin{enumerate}
    \item [label] item_1
    .....
    \item [label] item_n
```

\end{enumerate}

这个环境建立由 n 项列表项组成的列表，其中每个列表项以一个从 1 开始依次递增的数字作标识，\item 命令及 label 可选参数的含义同\itemize 环境。例如：

1. The first item of an itemize
2. The second item of an itemize

```
\begin{enumerate}
    \item The first item of an itemize
\item The second item of an itemize
\end{itemize}
```

3. 带关键词标识的列表环境

\begin{description}

\item [*keyword*] *item*$_1$

.....

\item [*keyword*] *item*$_n$

\end{description}

这个环境建立由 n 项列表项组成的列表，\item 命令的含义同前两种列表环境，每个列表项可由一个关键字用黑体标识，该关键字在相应\item 命令的可选参数 keyword 中给出（如果没有使用则本项不带标识）。例如：

first　The first item of an itemize
second　The second item of an itemize

```
\begin{description}
\item [first] The first item of an itemize
    \item [second] The second item of an
itemize
\end{description}
```

2.10.2　列表嵌套

各种列表环境可以嵌套，总的嵌套深度最多可达四层。如果同一种列表环境在不同层次上出现，除了关键字标识列表以外，内层所用的标识符号将依次变化。列表项的缩进量总是相对于相邻外层列表的左边界计算。对于圆点列表来说，嵌套时其列表项的标识符号变化的顺序（从外层到内层）依次是：

第一次出现时标识是一个黑点；

第二次出现时标识是一个长破折号；

第三次出现时标识是一个星号；

第四次出现时标识是一个小圆点。

编号列表 enumerate 环境嵌套时，其列表项的标识符号变化的顺序（从外层到内层）依次是：

第一次出现时标识是阿拉伯数字编号后跟一个圆点句号；

第二次出现时标识是小括号内加小写英文字母编号；

第三次出现时标识是小写罗马数字编号后跟一个圆点句号；

第四次出现时标识是大写英文字母编号后跟一个圆点句号。

例如：

```
\begin{itemize}
```

```
    \item First item in first-layer \textbf{itemize}.
    \begin{enumerate}
        \item First item in second-layer list and first-layer \textbf{enumerate}.
        \begin{itemize}
          \item First item in third-layer list and second-layer \textbf{itemize}.
          \begin{enumerate}
            \item First item in fourth-layer list and second-layer \textbf{enumerate}.
            \item Second item in fourth-layer list and second-layer \textbf{enumerate}.
          \end{enumerate}
          \item Second item in third-layer list and second-layer \textbf{itemize}.
        \end{itemize}
        \item Second item in second-layer list and first-layer \textbf{enumerate}.
    \end{enumerate}
    \item Second item in first-layer \textbf{itemize}.
\end{itemize}
```

将产生如下的输出结果：

- First item in first-layer **itemize**.
  1. First item in second-layer list and first-layer **enumerate**.
     - First item in third-layer list and second-layer **itemize**.
       (a) First item in fourth-layer list and second-layer **enumerate**.
       (b) Second item in fourth-layer list and second-layer **enumerate**.
     - Second item in third-layer list and second-layer **itemize**.
  2. Second item in second-layer list and first-layer **enumerate**.
- Second item in first-layer **itemize**.

使用\item 命令中的可选参数 label，可以改变 itemize 列表或 enumerate 列表中的列表项标识，如将 label 设为+，那么标识就变为+，而将 label 设为 2.1:，则标识就变为 2.1:。对于 enumerate 环境，如果使用了 label 参数，就意味着对应的编号记数器不再能够自动增加，用户必须手工改变它。

LaTeX 为 itemize 和 enumerate 环境各自四种缺省标识定义了总共 8 个命令：\labelitemi, \labelitemii, \labelitemiii, \labelitemiv 依次表示 itemize 环境第一层到第四层的列表项标识；而\labelenumi, \labelenumii, \labelenumiii, \labelenumiv 依次表示 enumerate 环境第一层到第四层的列表项标识。另外 LaTeX 还为 enumerate 列表环境定义了四个编号计数器 enumi, enumii, enumiii 和 enumiv，依次用于四层编号列表。我们可以使用\renewcommand 命令对这些标识命令重新定义，从而可以很容易地改变列表项缺省标识，不过改变 enumerate 列表的标识大都是改变其编号计数器的显示方式（有关计数器命令参见 2.1.2 节）。例如：

```
\renewcommand{\labelitemiii}{+}
\renewcommand{\labelenumii}{\arabic{enumii}.)}
```

第一条命令将把 labelenumii 所代表的标识定义为用阿拉伯数字显示的记数器 enumii 的值

后跟 '.' 使用这种方法时，enumerate 列表所有层次的编号都可以改变，有时编号甚至可以包含不止一个记数器，如控制序列

\renewcommand{\labelenumii}{\Alph{enumi}.\arabic{enumii}}

会使得每当在第二层调用\item 命令时使用大写字母输出的记数器 enumi 的值后跟阿拉伯数字形式的计数器 enumii 的值，即诸如 A.1, A.2,..., B.1, B.2 等等。

如果要使新的缺省标识适用于整个文档，就应当把\renewcommand 命令放在导言区，否则它就只对其所出现的那个环境内有效。

2.10.3 通用列表环境

1. 一般列表(list)环境

\begin{list}{*default-labeling*}{*spacing*}
　　\item [$label_1$] $item_1$
　　.....
　　\item [$label_n$] $item_n$
\end{list}

该环境定义更具一般意义的列表环境。必选参数 default-labeling 表示当\item 命令定义列表项时没有给出可选的标识参数，则使用 default-labeling 所代表的标识符号，该符号被放在一个文本盒子(我们称之为标识盒)中。default-labeling 参数可以也经常包含别的 LaTeX 命令。必选参数 spacing 包含改变列表项中各种间距的命令，如果我们想使用系统缺省设置，该参数可以置空（即使用{}而不能省略这对花括号）。

2. 与 list 环境相关的间距命令

这里我们介绍的长度命令影响 list 环境中列表项及其标识盒周围的空白情况，可以用 setlength 命令重新设置。有些长度的设置应当在 list 环境调用时的 spacing 参数中进行，对此我们将明确指出，凡是没有明确指出的命令，都可以在进入 list 环境之前进行全局性设置。另外，在设置垂直长度命令时，长度值最好都使用橡皮长度。各种间距的物理意义见图 2.1 所示。

(1) \itemindent 命令

列表项标识盒的左边界距 list 环境左边界之间的水平间隔（即列表项标识的缩进量），通常取其缺省值 0pt 。其值只能在 list 环境的 spacing 参数中设置，因此它的作用域也仅限于所定义的列表环境内部。

(2) \itemsep 命令

不同列表项之间的额外的垂直间隔，因此相邻列表项之间的实际间隔为\itemsep 加上段间间隔\parsep 。其值只能在 list 环境的 spacing 参数中设置，因此它的作用域也仅限于所定义的列表环境内部。

(3) \labelsep 命令

标识盒右边界距列表项文本显示区域左边界之间的水平间隔。虽然这个命令也可以在 list 环境外部进行全局性设置，但是这种设置也只影响第一层 list 环境。

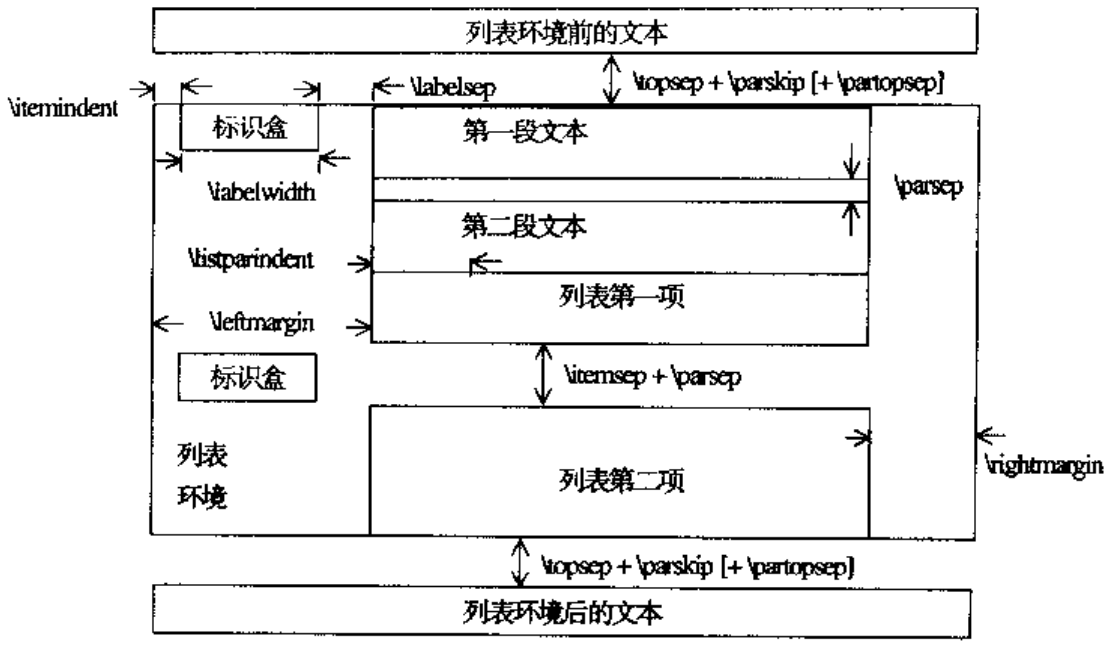


图 2.1 list 环境中的间隔命令及其含义

(4) \labelwidth 命令

列表项标识盒的宽度。

(5) \leftmargin 命令

列表项文本显示区域的左边界离其所属的 list 环境左边界之间的水平间隔。

(6) \listparindent 命令

列表项中每段的第一行行首相对于列表项文本显示区域的左边界所缩进的水平距离，一般都使用其缺省值 0pt（不缩进）。其值只能在 list 环境的 spacing 参数中设置，因此它的作用域也仅限于所定义的列表环境内部。

(7) \parsep 命令

代表在同一个列表项中不同段落之间的垂直间隔。其值只能在 list 环境的 spacing 参数中设置，因此它的作用域也仅限于所定义的列表环境内部。

(8) \rightmargin 命令

列表项文本显示区域的右边界离其所属的 list 环境右边界之间的水平间隔。缺省值为 0pt 。

(9) \partopsep 命令

当 list 环境与前后的文本之间存在空行时，list 环境与这些文本之间将额外多空出的垂直空白大小。因为空行不一定存在，所以在图 2.1 中我们将该命令放入一对中括号内。另外，图中的\parskip 代表文本状态下的段间间隔，即系统把 list 环境视为新的独立的一段。

(10) \topsep 命令

表示 list 环境与其前后的其他环境之间的垂直间隔。其值只能在 list 环境的 spacing 参数中设置，因此它的作用域也仅限于所定义的列表环境内部。

下面是一个 list 环境的应用示例：

```
\newcounter{fig}
\begin{list}{\bfseries\upshape Figure \arabic{fig}:}
     {\usecounter{fig}
     \setlength{\labelwidth}{2cm}\setlength{\leftmargin}{2.6cm}
     \setlength{\labelsep}{0.5cm}\setlength{\rightmargin}{1cm}
     \setlength{\parsep}{0.5ex plus0.2ex minus0.1ex}
     \setlength{\itemsep}{0ex plus0.2ex} \slshape}
     \item Page format with head, body, and foot, showing the
          meaning of the various elements involved.
     \item Format of a general list showing its elements.
     \item A demonstration of some of the possibilities for
          drawing pictures with \LaTeX.
\end{list}
```

其排版结果为

Figure 1: *Page format with head, body, and foot, showing the meaning of the various elements involved.*

Figure 2: *Format of a general list showing its elements.*

Figure 3: *A demonstration of some of the possibilities for drawing pictures with LaTeX*

3. 自定义列表环境

有时候我们需要反复使用一种精心设计的 list 环境，这时可以使用\newenvironment 命令定义一个新的列表环境。在定义过程中，我们可以设置必要的列表环境间距的值。例如上面的应用示例中的环境可以如下定义：

```
\newenvironment{figlist}{\begin{list}
     {\bfseries\upshape Figure \arabic{fig}:}
     {\usecounter{fig}\setlength{\labelwidth}{2cm}\setlength{\leftmargin}{2.6cm}
     \setlength{\labelsep}{0.5cm}\setlength{\rightmargin}{1cm}
     \setlength{\parsep}{0.5ex plus0.2ex minus0.1ex}
     \setlength{\itemsep}{0ex plus0.2ex} \slshape}
     \item[]}
     {\end{list}}
```

4. list 列表的嵌套

与 2.10.1 节中所介绍的列表环境相似，list 列表也可以嵌套，且最大层数可以到 6 层。另外 list 列表也可以同其他类型的列表嵌套。内层的列表环境相对于外层的列表环境都要向右缩进\leftmargin 大小的距离。

LaTeX 为可能处于 6 种不同层次的 list 列表提供了 6 个表示\leftmargin 大小的命令，依次为：\leftmargini, \leftmarginii, \leftmarginiii, \leftmarginiv, \leftmarginv 和\leftmarginvi。这些命令都必须在进入所对应的 list 列表环境之前加以定义，不能在 list 环境的 spacing（间隔）参数内部设置（此时只能设置\leftmargin）。如果某个 list 环境没有在间隔参数内明确设置\leftmargin 值，则根据它所在的层数缺省设置为\leftmargini 到\leftmarginvi 中的一个值。

5. 简洁列表(trivlist)环境

```
\begin{trivlist}
    .....
\end{trivlist}
```

该环境定义了一种“平凡的(trivial)”列表，这种列表可以看作是 list 列表的一种简化结果：

```
\begin{list}{}{\setlength {\leftmargin}{0pt}
               \setlength {\labelwidth}{0pt}
               \setlength {\itemindent}{0pt}
               \setlength {\listparindent}{\parindent}
               \setlength {\parsep}{\parskip}}
                 .....
\end{list}
```

LaTeX 内部实现时，有些别的环境就是用 trivlist 环境定义的，如 center 环境为

```
\begin{trivlist} \centering \item[]
    text
\end{trivlist}
```

第三章　增强排版效果

3.1　字型和字号

LaTeX 按照文档的逻辑结构（如节、脚注等等）选择合适的字型和字号（合称字体）。

3.1.1　设置字型

有时候我们希望人工改变字型，为此可以使用表 3.1 中的命令，同一表格单元中的两条命令等价。

表 3.1　　设置字型命令

命　令	字　型	命　令	字　型
\textrm \rmfamily	roman	\textsf \sffamily	sans serif
\texttt \ttfamily	typewriter	\textbf \bfseries	加黑
\textmd \mdseries	中等浓度（缺省值，与\textbf 相反）	\textit \itshape	斜体
\textup \upshape	正体（缺省值，与\textsl 相反）	\textsc \scshape	小体大写字母
\textsl \slshape	小斜体	\textnormal \normalfont	主文档标准字型
\emph	强调字型（\textit 和\textrm 间切换）		

这些命令的使用方式形如\textit{text}，作用范围为 text 文本中下一条设置字型命令或 text 文本末尾。设置字型命令有时候可以叠加，如\sffamily\bfseries 将设置加黑的 sans serif 字型。还可以使用字型命令的环境形式，如\begin{ttfamily}...\end{ttfamily}。

3.1.2　设置字号

表 3.2 中列出了设置不同大小字号的命令。每一种字体的实际尺寸取决于文档的类别和可选参数值，不过表 3.2 中的字号大小按照命令的从上到下、从左到右逐渐增大。

为了配合设置字号的命令，花括号扮演了重要的角色。花括号用来建立一个一个的组(group)，组限制了大多数 LaTeX 命令的作用范围。例如：

He likes large and small letters.

```
He likes {\LARGE large and
{\small small} letters}.
```

LaTeX2e 的一个重要的特性是字体属性之间是相互独立的，也就是说尽管你可以用设置字体或字号的命令改变字体属性，但是以前设置的 bold 字体或 slant 字体的属性依然保持不变。对于从头学起的读者来说这一点似乎是显而易见的，但是对于过去熟悉 LaTeX2.09 的读者来说就不是那么显然了。

表 3.2 设置字体尺寸命令

命 令	字 样	命 令	字 样
\tiny	tiny font	\large	large font
\scriptsize	very small font	\Large	larger font
\footnotesize	quite small font	\LARGE	very large font
\small	small font	\huge	huge
\normalsize	normal font	\Huge	largest

如果段落结束时仍处于字号命令的作用范围之内，也即字号命令对应的右花括号没有早在段落结束前出现，则字号命令也会改变行间距。注意下面两段例子中\par 命令的位置不同所带来的不同效果。

Don't read this! It is not

true. You can believe me!

```
{\Large Don't read this! It is not

    true. You can believe me!\par}
```

This is not true. But remember I am a liar.

```
{\Large This is not true.
    But remember I am a liar.}\par
```

3.1.3 底层字体命令

这些命令主要用于书写宏和包，下面列出的是其中常用的一些命令。

1. \fontencoding 命令

\fontencoding{*enc*}

这条命令选定字体编码方法，有效的方法有 OT1 和 T1 两种。

2. \fontfamily 命令

\fontfamily{*family*}

这条命令选择字体家族 family，下面是合法字体家族中的几种。

cmr: Computer Modern Roman

cmss: Computer Modern Sans Serif

cmtt: Computer Modern Typewriter

3. \fontseries 命令

\fontseries{*series*}

这条命令选择字体修饰 series，下面是部分合法的 series 取值（可以组合）：

m: 中等浓度（正常浓度）

b: 加黑

c: 紧缩

bc: 紧缩加黑

bx: 加宽加黑

4. \fontshape 命令

\fontshape{*shape*}

这条命令选择字体形状 shape，合法的 shape 取值如下：

n: 正体(\upshape)

it: 斜体(\itshape)

sl: 小斜体(\slshape)

sc: 小体大写字母(\scshape)

ui: 正体大斜体(\upshape\itshape)

ol: 大纲体

其中最后两种形状对大多数字体家族都不可用。

5. \fontsize 命令

\fontsize{*size*}{*skip*}

选择字号大小，第一个必选参数 size 指定要切换成的字号大小，第二个必选参数 skip 相当于\baselineskip 命令的参数，两个参数的缺省单位都是 pt，根据经验 skip 应当是 size 的 1.2 倍。

6. \selectfont 命令

\selectfont

刚才介绍的几条字体改变命令在遇到这条命令之前并不起作用。

7. \usefont 命令

\usefont{*enc*}{*family*}{*series*}{*shape*}

可以看作一条综合命令，其作用相当于使用相应的参数依次调用\fontencoding、\fontfamily、\fontseries 和\fontshape 命令，最后调用\selectfont 命令。

3.2 控制空白

3.2.1 行间距

如果你想拉大行与行之间的距离的话，可以在文档的导言区（介于\documentclass 命令和\begin{document}命令之间的区域）使用下面这条命令改变行间距的值：

\linespread{*factor*}

其中 factor 参数代表行间距放大的比例，例如\linespread{1.3}定义的行间距为 1.5 倍行距，而\linespread{1.6}定义的行距为 2 倍行距。正常情况下行间距并不拉大，因此 factor 参数的缺省值为 1。

3.2.2 段落格式

LaTeX 中有两个参数影响段落的布局 在输入文档的导言区中使用类似于下面两行文本的方法，可以改变段落的外观：

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

这两行文本增加两段之间的距离，同时将段落的缩进设置为 0。在欧洲大陆，段与段之间往往留一定的空白同时段首并不缩进，但是要知道这种设置同样要影响到目录列表，因而目录中的每一行就要变得稀疏了，因为目录表中的每一行实际上是被当作一段来对待的。

如果你想让原本不缩进的段落缩进，可以在段的开头部分使用\indent 命令，显然这条命令只能在\parindent 没有被设置为 0 的情况下才能起作用。为了设置每一节中开头第一段的缩进量，可以使用"tools"系列工具中的 indentfirst 包。

为了创建非缩进段落，可以在段落中将下面的命令作为头条命令：

```
\noindent
```

尤其当你不用分节命令而直接开始文档体时，这条命令更显方便。

3.2.3 水平间距

LaTeX 自动确定单词与单词之间、句子与句子之间的距离。为了增加水平间距，可以使用下面这条语句：

```
\hspace{length}
```

其中 length 参数最简单的格式就是一个数值加上一个距离单位。如果你希望即使这条命令所在位置落在行首或行尾也一样要保持指定大小的空白，则要使用\hspace*命令来代替\hspace 命令。下面是一个例子：

```
This      is a space of 1.5 cm.      This\hspace{1.5cm}is a space    of 1.5 cm.
```

3.2.4 垂直间距

LaTeX 自动确定段落与段落之间、节与节之间、小节与小节之间的距离。如有必要，可以使用下面的命令设置段与段之间的垂直距离：

```
\vspace{length}
```

其中 length 参数最简单的形式就是一个数字加上一个距离单位。通常这条命令应当在两个空行之间使用。如果一页顶部或底部也要保留指定长度的空白，应当使用该命令的星号版\vspace*代替它。使用相连的\stretch 命令和\pagebreak 命令可以对一页的最后一行文本进行排版，或者使文本在垂直方向上向页面中央居中对齐。

同一段落中或表格内部两行之间的距离可以由下面的命令指定：

```
\\[length]
```

其中 length 参数的意义同\vspace 命令中的 length 参数。

下面的 3 条命令可以在垂直方向上跳过不同的距离：

```
\bigskip
```

\medskip

\smallskip

\bigskip 命令的作用相当于\vspace{\bigskipamount}命令的作用，\medskip 命令的作用相当于\vspace{\medskipamount}命令的作用，\smallskip 命令的作用相当于\vspace{\smallskipamount}命令的作用。其中分别跳过的距离 bigskipamount，medskipamount 和 smallskipamount 均取决于文档类别。

3.3 文本盒子

1.\makebox 命令

\makebox[*width*][*position*]{*text*}

这条命令创建一个宽度恰好能容纳 text 文本的盒子。可选参数 width 可给出盒子的宽度；可选参数 position 可给出盒子内部 text 的位置，position 可以取下面这些值：

c: 居中（缺省值）

l: 左对齐

r: 右对齐

s: 从左边界拉伸到右边界，只有text中含有可拉伸的空格时这种选择才能有作用。

2. \mbox 命令

\mbox{*text*}

这条命令创建一个宽度仅能容纳 text 参数指定的文本的盒子，可以避免文本被线条割裂的现象。

3. \fbox 命令

\fbox{*text*}

这条命令同\mbox 命令的作用完全相同，只是在创建的盒子四周放一个方框。

4. \framebox 命令

\framebox[*width*][*position*]{*text*}

这条命令同\makebox 命令的作用完全相同，只是在创建的盒子四周放一个方框。同时这条命令创建一个标尺，标尺的厚度由\fboxrule 指定，标尺离盒子中内容的距离由\fboxsep 指定。

5. \newsavebox 命令

\newsavebox{*cmd*}

声明一个叫做 cmd 的柜子(bin)以存放盒子，cmd 必须是未被定义过的命令名称。

6. \sbox 命令

\sbox{*cmd*}{*text*}

这条命令对盒子中的文本进行排版，但是不打印输出结果盒子，而是将结果存于一个柜子 cmd 中，cmd 必须已经由\newsavebox 命令申明过。

7. \savebox 命令

\savebox{*cmd*}[*width*][*pos*]{*text*}

这条命令同\makebox 命令一样排版盒子中的文本 text，只不过不打印输出结果盒子，

而是把结果存于柜子 cmd 中，cmd 必须是用\newsavebox 命令申明过的。

8. \usebox 命令

\usebox{*cmd*}

打印输出最近用\savebox 命令存入柜子 cmd 的盒子。

9. lrbox 环境

\begin{lrbox}{*cmd*} *text* \end{lrbox}

这是\sbox 命令的环境版，环境中的 text 被存于 cmd 中，cmd 必须用\newsavebox 申明过.

10. \parbox 命令

\parbox[*position*][*height*][*inner-pos*]{*width*}{*text*}

这个命令创建一个段落盒子，其内容是段落模式的文本。其中必选参数 width 指定盒子的宽度，必选参数 text 给出盒子中的文本。

LaTeX 将使盒子的中心线与文本行的中央对齐（即文本水平居中）。可选参数 position 允许你指定盒子中的文本是顶端对齐还是底端对齐（缺省是顶端对齐）。如果没有给出可选参数 height，盒子的高度就等于文本的自然高度。可选参数 inner-pos 控制文本在盒子中的排放，如果没有给出该参数，系统就使用 position 参数来确定排放位置，position 可以取下列这些值：

t: 文本靠盒子的顶部放置

c: 文本垂直居中放置

b: 文本靠盒子的底部放置

s: 文本垂直拉伸，只有当text中含有可垂直拉伸的空格时这种选择才能起作用。

\parbox命令用于包含很少的文本，一般情况下也不要在段落盒子中使用任何创建段落的环境。如果要处理较大的文本块或者含有段落环境的文本，应当使用minipage环境，有关minipage环境参见2.2.6节。

11. \raisebox 命令

\raisebox{*distance*}[*extend-above*][*extend-below*]{*text*}

这条命令用于升高或降低文本 text。第一个必选参数 distance 指定升高的幅度（负值表示降低），文本按 LR 模式处理。第一个可选参数 extend-above 表示让文本超出顶线的上面多少距离，第二个可选参数 extend-below 表示让文本超出底线的下面多少距离。

12. \rule 命令

\rule[*raise-height*]{*width*}{*thickness*}

这条命令用于产生水平标尺线，各个参数的定义如下：

raise-height: 指定标尺线升高多少（可选）

width: 指定标尺的水平长度（必选）

thickness: 指定标尺的垂直厚度（必选）

3.4　其他修饰

3.4.1　定制页面布局

LaTeX 允许在\documentclass 命令中指定纸张大小，然后据此自动选择合适的页边距。但是有时我们或许对自动选择的值不满意，可以通过参数人为改变版面的布局。图 3.1 展示了可以改变设置的页面布局参数。

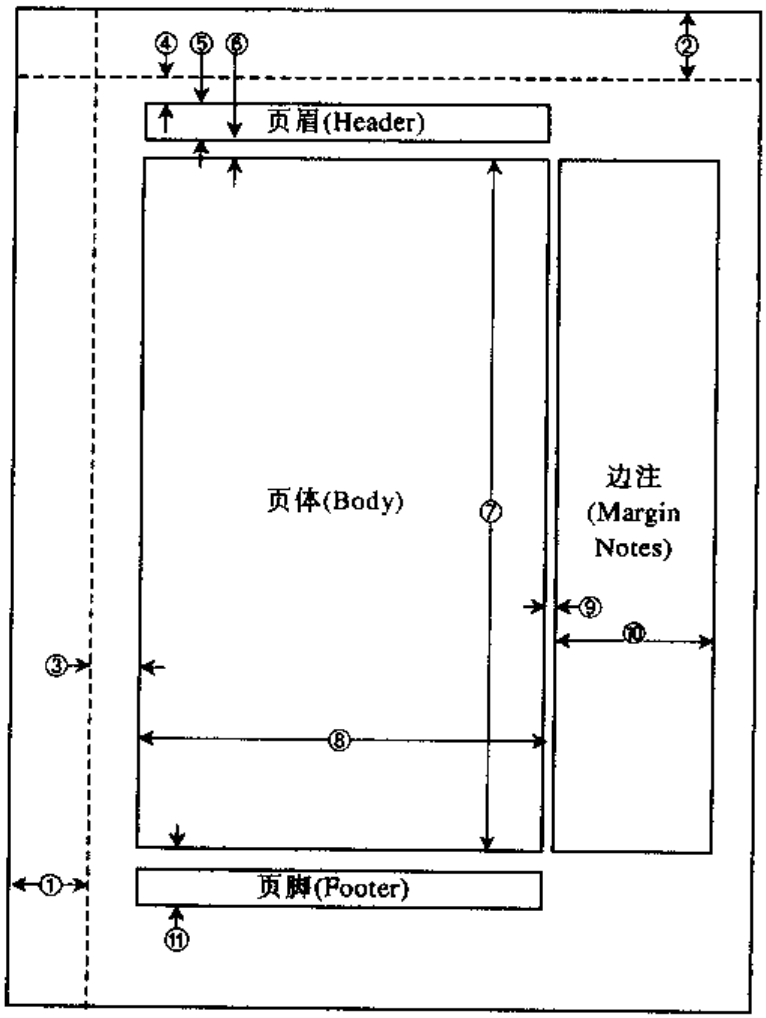


图 3.1　页面布局参数

[各种尺寸计算方法] ① 1in + \hoffset　② 1in + \voffset　③ \oddsidemargin = 22pt　④\topmargin=22pt
⑤ \headheight=13pt　⑥\headsep=19pt　⑦ textheight = 595pt　⑧ \textwidth = 360pt　⑨\marginparsep=7pt
⑩ \marginparwidth = 106pt　⑪ \footskip = 27pt

（图中未编号标注部分）\marginparpush = 5pt　　\hoffset = 0pt　　\voffset = 0pt
\paperwidth = 597pt　　\paperheight = 845pt

LaTeX 提供了两条命令以改变上述参数，通常在文档的导言区中使用。第一条命令给任何一个参数指派固定的值 length：

\setlength{*parameter*}{*length*}

其中 parameter 是一个页面布局参数的名称，length 为距离值。第二条命令可以使任何一个参数增加一个增量：

\addtolength{*parameter*}{*length*}

其中 parameter 和 length 的含义同上，只是 length 不再是绝对距离值，而是 parameter 的增量。实际上第二条命令更有用一些，因为使用它可以在现有页面设置（往往不需要知道各个具体参数的具体值）的基础上做相对调整。例如为了使总的文本宽度增加一个厘米，我们可以在文档导言区中使用下面的命令：

```
\addtolength{\hoffset}{-0.5cm}
\addtolength{\textwidth}{1cm}
```

3.4.2 参考文献布置

使用 thebibliography 环境可以产生一个参考文献列表，列表中的每一项用下面的命令开始：

\bibitem{*marker*}

其中 marker 用于标识该项参考文献。文档中在引用一个参考文献处使用下面的命令进行标注，其中的 marker 就是上面的 marker。

\cite{*marker*}

每条参考文献的编号工作是由系统自动完成的，\begin{thebibliography}命令后面的参数指定编号所占用的最大宽度值。例如：

```
Partl [1] has proposed, that . . .

1.1.1.1  Bibliography

[1] H. Partl: German TeX, TUG boat
    Vol. 9, No. 1 ('88)
```

```
Partl~\cite{pa} has
proposed, that \ldots

\begin{thebibliography}{99}
    \bibitem{pa} H.~Partl:
    \emph{German \TeX},
    TUGboat Vol.~9, No.~1 ('88)
\end{thebibliography}
```

3.4.3 建立关键字索引

许多书籍都提供了非常有用的关键字索引部分。在 LaTeX 和相应的支持程序（如有一种符合 8.3 文件名的程序叫 makeidx）的帮助下，生成关键字索引是一件十分容易的事。下面我们介绍一些基本的生成索引的 LaTeX 命令。

为了使 LaTeX 的索引功能起作用，文档的导言区中必须使用下面的语句激活（装入）makeidx 包：

\usepackage{makeidx}

同时导言区中还要有下面这条命令以使专门的索引命令能够使用：

\makeindex

关键字索引的列表项有下面的命令指定：

\index{*key*}

其中 key 是该索引项的标识。这条命令在文档中需要某个索引（由 key 参数标识）指向该处时使用。表 3.3 解释了 key 参数的语法格式及几个示例。

表 3.3　索引语句中 key 参数的格式及示例

示　例	索引项	说　明
\index{hello}	hello, 1	一般索引项
\index{hello!Peter}	Peter, 3	hello 下面的子索引项
\index{Sam@\textsl{Sam}}	*Sam* , 2	关键字带格式的索引项
\index{Lin@\textbf{Lin}}	**Lin**, 7	关键字带格式的索引项
\index{Jenny|textbf}	**Jenny, 3**	页码带格式的索引项
\index{Joe|textit}	Joe, *5*	页码带格式的索引项

LaTeX 处理输入文档文件时，每次碰到\index 命令，就将该命令的 key 连同命令所在处的页码（当前页码）一起写入一个特殊的索引文件，这个索引文件的主文件名与输入文件的主文件名相同，但是使用不同的后缀.idx。该.idx 文件以后可以用 makeindex 程序打开：

makeindex *filename.idx*

这条命令产生一个排好序的关键字索引文件，使用同样的主文件名和后缀名.ind。此时如果输入文档再次被 LaTeX 处理的话，排序的关键字索引列表可以用\printindex 命令插入到输入文档之中，插入位置就是该命令出现的位置。

随 LaTeX2e 发布了 showidx 包，可以将所有索引项打印在文档的左边边注位置。这一功能对于校对文档中各个索引的正确性的工作来说十分有用。

第四章 数学公式的排版

4.1 引 言

看起来不太复杂的数学公式，排起版来远没有那么容易。许多文字排版的方法都不适合于数学公式的排版工作。为了使排出来的数学公式更加美观，LaTeX 对文本排版风格进行了大量的改变，如数学环境中所有的字母都用斜体，"-"不再代表短破折号而代表减号，字符周围的空格处理也发生了变化（例如加号+周围留的空要比除号/周围的空稍大一些），等等。还有许多 LaTeX 的文字排版命令仍然可以在数学环境中使用，不过效果有些不同了，如\textbf 命令只影响公式中的字母和公式编号。{和}仍然是特殊字符，用于形成组将一段输入文档包围起来。

数学模式和段落模式（文本模式）有许多差别，例如数学模式下有下列特点：

1. 大多数的空格和换行命令不再有意义，因为数学表达式中的空格要么是系统根据表达式逻辑上导出的，要么要用特殊的命令指定，这些命令有\␣, \quad 和\qquad。例如：

$$\forall x\in \mathbf{R}: \qquad x^2 \geq 0 \qquad (4.1)$$

```
\begin{equation}
    \forall x \in \mathbf{R}:
    \qquad x^{2} \geq 0
    \end{equation}
```

2. 不允许出现空行，每个公式只有一段。

3. 每个字母都将被当作一个变量的名字并照此排版。如果在公式中你想排版一般文本（一般正体字体和正常空格），则必须将文本放在\textrm{...}命令的参数中。例如：

$$x^2 \geq 0 \text{ for all } \quad x\in \mathbf{R} \qquad (4.2)$$

```
\begin{equation}
    x^{2} \geq 0\qquad
    \textrm{for all }x\in\mathbf{R}
\end{equation}
```

大部分的数学模式命令只能对其后的一个字符起作用，如果想让数学模式命令作用于多个字符，必须用一对花括号将它们放进一个组内，如：

$$a^x + y \neq a^{x+y} \qquad (4.3)$$

```
\begin{equation}
    a^x+y \neq a^{x+y}
\end{equation}
```

4.2　数学环境

4.2.1　单行公式环境

文本段落中的数学符号可以在\(和\)之间、$和$之间或者\begin{math}与\end{math}之间输入。例如：

Add a squared and b squared to get c squared. Or using a more mathemat - ical approach: $c^2 = a^2 + b^2$

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or using
a more mathematical approach:
$c^{2}=a^{2}+b^{2}$
```

4.2.2　多行公式环境

1. displaymath 环境

```
\begin{ displaymath }
    equation
\end{ displaymath }
```

或

```
\[ equation \]
```

有时候我们需要排一个占多行的大数学公式，这时可以把公式包括在一个 displaymath 环境内部，\[和\]分别是\begin{displaymath}和\end{displaymath}的缩写形式。这个产生的数学公式没有编号，例如：

Add a squared and b squared to get c squared. Or using a more mathemat - ical approach:

$$c^2 = a^2 + b^2$$

And just one more line.

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or using
a more mathematical approach:
begin{displaymath}
c^{2}=a^{2}+b^{2}
\end{displaymath}
And just one more line.
```

2. equation 环境

使用这个环境排版的数学公式带有编号。一般情况下不要在公式的前面带有空行，这样公式会另起一段，比较难看。

使用\label 和\ref 命令可以在文本中引用公式，如：

$$\varepsilon > 0 \qquad (4.4)$$

From (4.4) we gather...

```
\begin{equation} \label{eq:eps}
    \epsilon > 0
\end{equation}
    From (\ref{eq:eps}) we gather
    \ldots
```

3. eqnarray 环境

```
\begin{eqnarray}
    line1 [\nonumber]\\
    ..........
\end{eqnarray}
```

这个环境用于生成方程组，每一行代表一个方程。如果一个方程使用了\nonumber 命令，表示该方程所在的行不产生编号（缺省时每个方程都产生编号）。如：

$$a1 = b1 + c1$$
$$a2 = b2 - c2 \qquad (2)$$

```
\begin{eqnarray}
    a1 & = & b1 + c1 \nonumber\\
    a2 & = & b2 - c2
\end{eqnarray}
```

这个环境还有一个星号版，eqnarray*环境将每个方程的编号压缩成整个方程组一个编号。

注意上述数学环境有着各自不同的缺省属性（如字符尺寸等），一行数学公式同其上下行的文本之间保持较为稀疏的距离。如果你想在数学公式中使用一些非数学模式下的文本，必须用\mbox 创建一个盒子放置这些文本。

4.3 数学符号表

本节将以表格的形式列举数学模式下可以使用的各类符号及其输入方法。

4.3.1 数学模式重音符号

见表 4.1 所示。

表 4.1 数学模式下的重音符号

$\hat{e}$	\hat{e}	$\check{e}$	\check{e}	$\tilde{o}$	\tilde{o}	$\acute{o}$	\acute{o}
$\grave{e}$	\grave{e}	$\dot{e}$	\dot{e}	$\ddot{e}$	\ddot{e}	$\breve{e}$	\breve{e}
$\bar{a}$	\bar{a}	$\vec{a}$	\vec{a}	$\widehat{A}$	\widehat{A}	$\widetilde{A}$	\widetilde{A}

4.3.2 希腊字母

1. 小写希腊字母

见表 4.2 所示。

表 4.2 小写希腊字母表

α	\alpha	β	\beta	γ	\gamma	δ	\delta	ϵ	\epsilon
ε	\varepsilon	ζ	\zeta	η	\eta	θ	\theta	ϑ	\vartheta
ι	\iota	κ	\kappa	λ	\lambda	μ	\mu	ν	\nu
ξ	\xi	π	\pi	ϖ	\varpi	ρ	\rho	σ	\sigma
ς	\varsigma	τ	\tau	υ	\upsilon	ϕ	\phi	φ	\varphi
χ	\chi	ψ	\psi	ω	\omega	o	o	ϱ	\varrho

2. 大写希腊字母

见表 4.3 所示。

表 4.3　　大写希腊字母表

Γ	\Gamma	Δ	\Delta	Θ	\Theta	Λ	\Lambda
Ξ	\Xi	Π	\Pi	Σ	\Sigma	Υ	\Upsilon
Φ	\Phi	Ψ	\Psi	Ω	\Omega	$\digamma$	\Digamma

4.3.3　二元关系符号

见表 4.4 所示，表中有些符号可以直接按原样输入（如 >, = 等），有些符号后有用逗号分隔的两条命令，表示使用任何一条命令均可。其中\sqsubset，\sqsupset 以及\Join 3 种符号使用时必须装入 latexsym 包。所有的关系符号输入命令前如果加上\not 命令，可以得到与其相反关系的符号。

表 4.4　　二元关系符号表

$<$	<	$>$	>	$=$	=	$\leq$	\leq, \le
$\geq$	\geq, \ge	$\equiv$	\equiv	$\ll$	\ll	$\gg$	\gg
$\doteq$	\doteq	$\prec$	\prec	$\succ$	\succ	$\sim$	\sim
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq	$\subset$	\subset
$\supset$	\supset	$\approx$	\approx	$\subseteq$	\subseteq	$\supseteq$	\supseteq
$\cong$	\cong	$\sqsubset$	\sqsubset	$\sqsupset$	\sqsupset	$\bowtie$	\Join
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\bowtie$	\bowtie	$\in$	\in
$\ni$	\ni, \owns	$\propto$	\propto	$\vdash$	\vdash	$\dashv$	\dashv
$\models$	\models	$\mid$	\mid	$\parallel$	\parallel	$\perp$	\perp
$\smile$	\smile	$\frown$	\frown	$\asymp$	\asymp	:	:
$\notin$	\notin	$\neq$	\neq, \ne				

4.3.4 二元运算符号

表 4.5 二元运算符号表

+	+	-	-	±	\pm	∓	\mp
◁	\triangleleft	·	\cdot	÷	\div	▷	\triangleright
×	\times	\	\setminus	⋆	\star	∪	\cup
∩	\cap	*	\ast	⊔	\sqcup	⊓	\sqcap
∘	\circ	∨	\vee, \lor	∧	\wedge, \land	•	\bullet
⊕	\oplus	⊖	\ominus	◇	\diamond	⊙	\odot
⊘	\oslash	⊎	\uplus	⊗	\otimes	○	\bigcirc
⨿	\amalg	△	\bigtriangleup	▽	\bigtriangledown	†	\dagger
⊲	\lhd	⊳	\rhd	‡	\ddagger	⊴	\unlhd
⊵	\unrhd	≀	\wr				

见表 4.5 所示，表中有些符号可以直接按原样输入（如 + 和 - ），有些符号后有用逗号分隔的两条命令，表示使用任何一条命令均可。其中\lhd，\rhd，\unlhd 以及\unrhd 4 种符号使用时必须装入 latexsym 包。

4.3.5 大运算符

大运算符见表 4.6 所示。

表 4.6 大运算符表

Σ	\sum	⋃	\bigcup	⋁	\bigvee	⨁	\bigoplus	∏	\prod
⋂	\bigcap	⋀	\bigwedge	⨂	\bigotimes	∐	\coprod	⨆	\bigsqcup
⨀	\bigodot	∫	\int	∮	\oint	⨄	\biguplus		

4.3.6　箭头符号

箭头符号见表 4.7 所示。其中\leadsto 符号只有当 latexsym 包装入后才能使用。

表 4.7　箭头符号一览表

←	\leftarrow, \gets	⟵	\longleftarrow	↑	\uparrow	→	\rightarrow, \to
⟶	\longrightarrow	↓	\downarrow	↔	\leftrightarrow	⟷	\longleftrightarrow
↕	\updownarrow	⇐	\Leftarrow	⟸	\Longleftarrow	⇑	\Uparrow
⇒	\Rightarrow	⟹	\Longrightarrow	⇓	\Downarrow	⇔	\Leftrightarrow
⟺	\Longleftrightarrow	⇕	\Updownarrow	↦	\mapsto	⟼	\longmapsto
↗	\nearrow	↩	\hookleftarrow	↪	\hookrightarrow	↘	\searrow
↼	\leftharpoonup	⇀	\rightharpoonup	↙	\swarrow	↽	\leftharpoondown
⇁	\rightharpoondown	↖	\nwarrow	⇌	\rightleftharpoons	⟺	\iff
⇝	\leadsto						

4.3.7　分隔定界符号

1.一般分隔定界符号

一般分隔定界符号见表 4.8 所示。

表 4.8　一般分隔定界符号表

(	(	)	)	↑	\uparrow	⇑	\Uparrow
[	[, \lbrack	]	], \rbrack	↓	\downarrow	⇓	\Downarrow
{	\{, \lbrace	}	\}, \rbrace	↕	\updownarrow	⇕	\Updownarrow
⟨	\langle	⟩	\rangle	|	|, \vert	‖	\\|, \Vert
⌊	\lfloor	⌋	\rfloor	⌈	\lceil	⌉	\rceil
/	/	\	\backslash	.	.		

2.大分隔定界符号

大分隔定界符号见表 4.9 所示。另外表 4.8 中绝大部分定界符号命令前面使用\big, \Big, \bigg 以及\Bigg 作为命令前缀，就可以得到尺寸越来越大的定界符号。

表 4.9　大分隔定界符表

⟮	\lgroup	⟯	\rgroup	⎰	\lmoustache	⎱	\rmoustache
⎟	\arrowvert	⦀	\Arrowvert	⎪	\bracevert		

4.3.8 其他杂类符号

表 4.10 杂类符号一览表

…	\dots	⋯	\cdots	⋮	\vdots	⋱	\ddots
ℏ	\hbar	ı	\imath	ȷ	\jmath	ℓ	\ell
ℜ	\Re	ℑ	\Im	ℵ	\aleph	℘	\wp
∀	\forall	∃	\exists	∃	\mho	∂	\partial
′	'	/	\prime	∅	\emptyset	∞	\infty
∇	\nabla	∆	\triangle	□	\Box	◇	\Diamond
⊥	\bot	⊤	\top	∠	\angle	√	\surd
♢	\diamondsuit	♡	\heartsuit	♣	\clubsuit	♠	\spadesuit
¬	\neg, \lnot	♭	\flat	♮	\natural	♯	\sharp

表 4.10 列出了一些用途广泛但又不好归类的许多符号，其中\mho，\Box 以及\Diamond 这 3 个符号只有当 latexsym 包装入后才能使用。还有几个符号不仅数学模式下可用，文本模式下也可以使用，我们称之为非数学符号，见表 4.11 所示。

表 4.11 非数学符号表

†	\dag	§	\S	©	\copyright
‡	\ddag	¶	\P	£	\pounds

4.4 数学环境中文本修饰

4.4.1 字符和字符串修饰命令

有时候我们需要在字母或若干字母上加上一些修饰，LaTeX 提供了如表 4.12 所列的修饰命令，表 4.12 中同时给出了每个命令的使用示例。

从表 4.12 中我们可以看出，\not 同别的修饰命令不同，它所修饰的内容（仅限单个字符）不是作为参数放在组中，而是跟在命令的后面。另外虽然\not 命令可以使用，但是它的修饰效果不太理想，同时\widehat 和\widetilde 命令在参数字符串比较长时效果也不好，为了解决这些问题，可以使用后面 4.5.5 节中介绍的空白控制方法。

表 4.12　字符（串）修饰命令及应用示例

$\hat{e}$	\hat{e}	$\widehat{easy}$	\widehat{easy}
$\tilde{o}$	\tilde{o}	$\widetilde{easy}$	\widetilde{easy}
$\check{e}$	\check{e}	$\breve{e}$	\breve{e}
$\acute{o}$	\acute{o}	$\grave{E}$	\grave{e}
$\bar{a}$	\bar{a}	$\vec{e}$	\vec{e}
$\dot{e}$	\dot{e}	$\ddot{E}$	\ddot{e}
$\not{e}$	\not e		

如果你想把一个字符串放到另一个字符串的头顶上，可以使用下面这条命令：

\stackrel{*top-text*}{*bottom-text*}

其中 top-text 代表放在上面的文字，bottom-text 代表放在下面的文字。top-text 将以 small 尺寸的字符紧靠在 bottom-text 上方。如：

$a \stackrel{def}{=} b + c$　　$ a \stackrel{def}{=} b + c $

这种叠放也可以使用 atop 宏来实现。

4.4.2　常见函数名控制序列

在数学模式下，LaTeX 通常将文字用数学斜体显示。使用下面列出的这些代表常见函数的控制序列，可以使函数名能够以正体的形式显示，同时 LaTeX 对于这些函数名所带的上下标进行检查，避免不适当的上下标符号出现：

```
\arccos  \arcsin  \arctan  \arg  \cos  \cosh  \cot
\coth  \csc  \deg  \det  \dim  \exp  \gcd
\hom  \inf  \ker  \lg  \lim  \liminf  \limsup
\ln  \log  \max  \min  \Pr  \sec  \sin
\sinh  \sup  \tan  \tanh
```

4.4.3　上下标符号

上下标符号分别由^和_符号引导。根据基字符的种类和风格，上下标符号要么显示在基字符的右上角或右下角，要么直接显示在基字符的上面或下面，比如基字符如果是英文字母的话，上下标均显示在基字符的右侧。例如：

F_2^3　　F_2^3

求和符号(\sum)的上下标缺省放置位置也是右侧，如：

$\sum_{i=1}^{n}$　　$\sum_{i=1}^{n}$

如果我们想把上下标放在求和符号的正上方和正下方，可以使用命令\limits，如：

$$\sum\limits_{i=1}^{n}$$ $\sum\limits_{i=1}^n$

4.4.4 上下画线修饰

1.上画线命令

\overline{*text*}

这条命令将在 text 所代表的文字上面画一条直线。

2.下画线命令

\underline{*text*}

这条命令将在 text 所代表的文字下面画一条直线，一般来说在同一行中不同时使用上下画线。

4.4.5 上下花括号修饰

1. 上花括号命令

\overbrace{*text*}[^{*upper-note*}]

这条命令将在 text 所代表的文字上面画一条横放的开口朝下的长花括号，可选参数 upper-note 为放在花括号上方的说明文字，大小同上标大小。

2. 下花括号命令

\underbrace{*text*}[_{*under-note*}]

这条命令将在 text 所代表的文字下面画一条横放的开口朝上的长花括号，可选参数 upper-note 为放在花括号下方的说明文字，大小同下标大小。

4.4.6 符号加黑

在 LaTeX 中要想得到加黑的符号不太容易，也许这是系统设计时故意所为，因为业余的排版人员经常过多地使用加黑处理功能。改变字体的命令\mathbf 可以对字母加黑，但是它是针对正体字，而数学符号通常都是斜体字。另外有一条\boldmath 命令，可是令人费解的是它只能在数学模式以外的环境中使用，不过它对特殊符号也同样有效。如：

$\mu, M \quad \mathbf{M} \quad$ **μ, M**

```
\begin{displaymath}
    \mu, M \qquad \mathbf{M} \qquad
    \mbox{\boldmath $\mu, M$}
\end{displaymath}
```

注意上面的例子中\boldmath 命令是放在 mbox 内部、数学环境外部使用的。另外它使 \mu 和 M 之间的逗号也加黑了，这往往不是我们的本意。

amsmath 包中包含的 amsbsy 包使得数学模式下各种符号的加黑工作变得容易得多。这个包提供一条\boldsymbol{text}命令，这条命令可以在数学模式下工作并将 text 文本（可以包含特殊字符）加黑。另外还提供了一条\pmb{text}命令，pmb 是英文“poor man's bold”的缩写，顾名思义，这条命令就是给那些习惯于过多使用加黑字体的“可怜人”用的，系

统将视情况忽略一些这种加黑命令　如：

$\mu, M \quad \mu, M \quad \mu, M$

```
\begin{displaymath}
    \mu, M \qquad
    \boldsymbol{\mu}, \boldsymbol{M}
    \qquad \pmb{\mu}, \pmb{M}
\end{displaymath}
```

从后面的例子中我们可以发现两条\boldsymbol 命令都是在数学环境内部使用的，并将它们的参数文本加黑了。另外注意两条\pmb 并没有达到加黑的效果。

4.5　数学公式的组成部分和间隔

每一个数学公式不论其多么复杂，在 LaTeX 中都可以分解成下列组成部分的有限次组合：基本符号、开方、分数、积分加上各种分隔定界符号。基本符号包括基本 ASCII 码符号、4.3 节中介绍的各种符号以及用 4.4 节中介绍的修饰命令修饰后形成的符号。下面我们分别介绍开方、分数和积分的输入方法。

4.5.1　开方

开方符号用\sqrt 输入，n 次开方用\sqrt[n]输入，开方符号的大小由 LaTeX 自动确定。如果开方符号不需要上面的横线面只需要左边的部分，可以使用\surd 命令。例如：

$\sqrt{x} \quad \sqrt{x^2+\sqrt{y}} \quad \sqrt[3]{2}$
$\sqrt{[x^2+y^2]}$

```
$\sqrt{x}$\qquad \sqrt{ x^{2}+\sqrt{y} }$
\qquad $\sqrt[3]{2}$\\[3pt]
$\surd[x^2 + y^2]$
```

4.5.2　分式与二项组合式

可以用下面的命令产生一个分式：

```
\frac{text1}{text2}
```

其中 text1 代表分子表达式，text2 代表分母表达式。例如：

$$\lim_{n\to 0}\frac{\sin x}{x}=1$$

```
\[\lim_{n \rightarrow 0}
\frac{\sin x}{x}=1\]
```

有时候对于比较短小的分数式使用斜线分隔分子与分母更好看一些，如 1/2 等等：

$1\frac{1}{2}$-hours

$\frac{x^2}{k+1} \quad x^{\frac{2}{k+1}} \quad x^{1/2}$

```
$1\frac{1}{2}$-hours
\begin{displaymath}
    \frac{ x^{2} }{ k+1 }\qquad
    x^{ \frac{2}{k+1} }\qquad
    x^{ 1/2 }
\end{displaymath}
```

有些数学公式的组成部分同分数式相似，也由上下两个式子构成，但是中间没有分号，这样的式子我们称之为二项组合式(binomial coefficients)。二项组合式的输入可以使用下面

两条命令中的一条：

text1 \choose *text2*

text1 \atop *text2*

它们的功能相似，都是把表达式 text1 放在表达式 text2 的上面，区别在于第一条命令给二项组合式的左右加上大的圆括号而第二条命令不加括号。例如：

$$\binom{n}{k} \qquad \begin{matrix} x \\ y+2 \end{matrix}$$

```
\begin{displaymath}
    {n \choose k}\qquad {x \atop y+2}
\end{displaymath}
```

4.5.3 积分与求和符号

积分符号用\int 命令产生，求和运算符用\sum 命令产生。定积分的上下界和求和的上下限如同上下标一样分别由^和_引导。如：

$$\sum_{i=1}^{n} \qquad \int_{0}^{\frac{\pi}{2}}$$

```
\begin{displaymath}
    \sum_{i=1}^{n} \qquad
    \int_{0}^{\frac{\pi}{2}} \qquad
\end{displaymath}
```

可能细心的读者发现，在 4.4.3 节中介绍上下标时我们曾经提到，求和符号的缺省上下标位置是在右侧，而上例中却在求和符号的垂直方向，原因是 4.4.3 节中求和符号处于 $...$定义的数学模式环境，而此处求和符号处于 displaymath 数学模式环境。

4.5.4 尺寸匹配的定界符

4.3.7 节中我们列出了所有 LaTeX 定界符号的输入方法，但是定界符号有其特殊性，它们的大小往往需要同其所包含的内容的高度相适应。那么我们如何调整定界符号的大小呢？基本上有下面两种方法。

1.自动调整方法

如果你在代表左边定界符号的命令（即开始定界）之前使用\left 命令，同时在相应的右边定界符号前（即结束定界）使用\right 命令，则 LaTeX 将自动调整左右定界符号的高度。注意\left 和\right 命令必须成对出现，有时候你可能只想使用左定界符而不用右定界符，你必须使用"\right."（右定界符不可见命令）结束定界，相似地，"\left."为左定界符不可见命令，用于只需要右定界符的场合。例如：

$$1+\left(\frac{1}{1-x^2}\right)^3\Bigg]$$

```
\begin{displaymath}
    \left.1 + \left( \frac{1}{ 1-x^{2} }
    \right) ^3\right\]
\end{displaymath}
```

2.手工调整方法

有时候我们有必要手工指定数学定界符的尺寸，这时可以使用\big, \Big, \bigg 和\Bigg 作为大部分定界符命令的前缀，例如：

$$((x+1)(x-1))^2$$

$$((((\quad))))^{\|\|\|\|}$$

```
$\Big( (x+1) (x-1) \Big) ^{2}$\\
$\big(\Big(\bigg(\Bigg($\quad
$\big\)\Big\)\bigg\)\Bigg\)$\quad
$\big\|\Big\|\bigg\|\Bigg\|$
```

但是要注意的是如果公式中使用了改变字体尺寸的命令，上述命令产生的大尺寸的定界符的尺寸也会发生变化

4.5.5　公式中的间隔

如果LaTeX为公式中间自动留出的空白不合适，你可以在公式中插入特殊的空格命令，以调整公式的外观。下面是一些较为重要的空格命令。

1. \, 命令：留一点小小的空白；

2. \␣ 命令：留中等大小的空格，其中␣代表一个空格字符；

3. \quad 和\qquad 命令：留一个较大距离的空白（差不多相当于通常按下 Tab 键所跳过的距离）；

4. \! 命令：压缩空白位置。

下面是一个例子：

$$\iint_D g(x,y)\mathrm{d}x\mathrm{d}y$$

instead of

$$\int\int_D g(x,y)\mathrm{d}x\mathrm{d}y$$

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
    \int\!\!\!\int_{D} g(x,y)
    \, \ud x\, \ud y
\end{displaymath}
instead of
\begin{displaymath}
    \int\int_{D} g(x,y)\ud x \ud y
\end{displaymath}
```

4.6　垂直对齐的数学元素

4.6.1　矩阵、行列式和分段函数

为了排版矩阵，可以使用 array 矩阵环境。矩阵环境有些像表格环境，只是其中的内容均处于数学模式。矩阵环境的使用有如下形式（以产生 3x3 矩阵为例，其余类推）：

```
\left( \begin{array}{lrc}
text11 & text12 & text13\\
text21 & text22 & text23\\
text31 & text32 & text33
\end{array} \right)
```

其中 text11,...,text33 代表矩阵中的元素，逐行从左到右排列，每一行用\\命令或\cr命令结束。\left(和\right)是为了产生矩阵两边的大括号。必选参数 lrc 中的每个字符代表了矩阵中相应

列其元素的垂直对齐方式，c 指定居中对齐、l 指定左对齐，r 指定右对齐。如 ccc 表示三列都是左对齐，而 lcr 表示第一列左对齐，第二列居中对齐，第三列右对齐等等。例如：

$$\mathbf{X}=\begin{pmatrix} x_{11} & x_{12} & \cdots \\ x_{21} & x_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

```
\begin{displaymath}
    \mathbf{X} =
    \left( \begin{array}{ccc}
        x_{11} & x_{12} & \ldots \\
        x_{21} & x_{22} & \ldots \\
        \vdots & \vdots & \ddots
    \end{array} \right)
\end{displaymath}
```

矩阵环境也可以用来排版分段函数，只不过不要右边的分节符罢了。另外分段函数我们一般把它当作两列矩阵看待，一列表示函数定义，另一列表示分段区间，一般来说这两段都用左对齐方式。例如：

$$y=\begin{cases} a & \text{if } d>c \\ b+x & \text{in the morning} \\ l & \text{all day long} \end{cases}$$

```
\begin{displaymath}
    y = \left\{ \begin{array}{ll}
        a & \textrm{if $d>c$}\\
        b+x & \textrm{in the morning}\\
        l & \textrm{all day long}
    \end{array} \right.
\end{displaymath}
```

如果你要排版一组方程，这时可以使用 eqnarray 和 eqnarray*环境代替使用 equation 环境。eqnarray 环境中的每一行都有一个公式编号，eqnarray*环境中的公式都没有编号。例如：

$$f(x)=\cos x \quad (4.5)$$

$$f'(x)=-\sin x \quad (4.6)$$

$$\int_0^x f(y)dy=\sin x \quad (4.7)$$

```
\begin{eqnarray}
    f(x) & = & \cos x \\
    f'(x) & = & -\sin x \\
    \int_{0}^{x} f(y)dy &
    = & \sin x
\end{eqnarray}
```

注意这个例子中中间列（等号）两边的空白都太大了一些，为了减小这些空白，可以使用\setlength\arrayclosep{2pt}命令，4.6.2 节中的公式 4.8 将应用这一命令。

4.6.2 分行长公式

对于很长的公式，LaTeX 不能自动地将它们断成整洁的数行，作者必须指定从公式的什么地方断开，换行后又要缩进多少等等。为了达到将长公式分行的目的，下面是两种最常用的办法。

```
{\setlength\arraycolsep{2pt}
\begin{eqnarray}
    \sin x & = & x -\frac{x^{3}}{3!}
    +\frac{x^{5}}{5!}-{}
    \nonumber\\
     & & {}-\frac{x^{7}}{7!}+{}\cdots
\end{eqnarray}}
```

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots \qquad (4.8)$$

```
\begin{eqnarray}
    \lefteqn{ \cos x = 1
    -\frac{x^{2}}{2!} +{} }
    \nonumber\\
    & & {}+\frac{x^{4}}{4!}
        -\frac{x^{6}}{6!}+{}\cdots
\end{eqnarray}
```

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots \qquad (4.9)$$

上面例子中的\nomumber 命令使 LaTeX 不在公式未结束的行的行末产生公式编号。使用上述例示的两种方法都很难将一个多行公式垂直对齐，amsmath 包提供了一些更为有效的方法。

4.7　数学字体

工作在数学方式下时，用户可以使用字体改变命令临时性地退出数学状态并输入一些正常文本。如果你想切换到另一种数学排版字体的话，还有专门的命令集，参见表 4.13。另外\mathversion{bold}可以用来设置公式中的加黑字母或符号，\mathversion{normal}可以用来恢复公式中字型的缺省浓度。

表 4.13　　数学字体一览表

命　令	输　入	输　出
\mathcal{...}	$\mathcal{B}=c$	$\mathcal{B}=c$
\mathrm{...}	$\mathrm{k}_2	k_2
\mathbf{...}	$\sum x=\mathbf{v}$	$\sum x=\mathbf{v}$
\mathsf{...}	$\mathsf{G\times R}$	$\mathsf{G\times R}$
\mathtt{...}	$\mathtt{L}(b,c)$	$\mathtt{L}(b,c)$
\mathnormal{...}	$\mathnormal{R_{19}}\neq R_{19}$	$\mathnormal{R_{19}}\neq R_{19}$
\mathit{...}	$\mathit{ffi}\neq ffi$	$\mathit{ffi}\neq ffi$

数学模式下 LaTeX 根据上下文选择字体的字号，比如上标部分使用较小的字号。如果你想在公式中使用\textrm 命令插入一些 roman 字体的文本，那么这些文本的字号将不会根据上下文自动适应，因为\textrm 命令使得工作模式临时转变到文本模式。使用表 4.13 中列出的\mathrm 命令虽然能避免上述问题，但是要注意这条命令只适合于较短的文本，同时

空白控制功能不再有效，各种重音符号也不能使用。不过 AMS-LaTeX 包已经将\textrm 命令扩充，使之能够根据数学环境中的上下文自动变化字号。

下面是一个例子：

$2^{\mathrm{nd}}\ 2^{\mathrm{nd}}$ (4.10)

```
\begin{equation}
2^\textrm{nd} \quad
2^\mathrm{nd}
\end{equation}
```

有时候我们需要告诉 LaTeX 字体的具体尺寸，这时可以使用下面 4 条在数学模式下通过改变字体风格改变字号大小的命令，后面括号内的是文本示例：

\displaystyle (123)

\textstyle (123)

\scriptstyle (123)

\scriptscriptstyle (123)

上述改变字体风格的命令同样影响定界符的显示尺寸，例如下面例子中的一对方括号就比标准的命令\left[\right]所产生的方括号稍小一些：

$$\mathrm{corr}(X,Y)=\frac{\displaystyle\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\displaystyle\Big[\sum_{i=1}^{n}(x_i-\overline{x})^2\sum_{i=1}^{n}(y_i-\overline{y})^2\Big]^{1/2}}$$

```
\begin{displaymath}
\mathop{\mathrm{corr}}(X,Y)=
\frac{\displaystyle
\sum_{i=1}^n(x_i-\overline x)
(y_i-\overline y)}
{\displaystyle\biggl[
\sum_{i=1}^n(x_i-\overline x)^2
\sum_{i=1}^n(y_i-\overline y)^2
\biggr]^{1/2}}
\end{displaymath}
```

4.8 变量描述文本和自定义定理环境

4.8.1 变量描述

在有些公式的后面你可能还想加上一段文字用以说明公式中的变量的含义，如下面的例子中说明了变量 a, b, c 的意义：

$$a^2+b^2=c^2$$

```
\begin{displaymath}
  a^2+b^2=c^2
\end{displaymath}
```

Where: a, b – are adjunct to the right angle of a right - angled triangle.

c – is the hypotenuse of the triangle.

```
{\settowidth{\parindent}
{Where:\ }
```

```
\makebox[0pt][r]
{Where:\ }$a$, $b$ -- are
adjunct to the right angle
of a right-angled triangle.

$c$ -- is the hypotenuse of
the triangle.}
```

如果你经常排这种变量描述文本，可以使用\newenvironment 命令自己定义一种新的环境，用以创建专门放置变量描述文本的环境。

4.8.2　定理结构

当我们书写数学文档时，可能经常遇到引理、定理、定义、公理等等诸如此类的结构，我们统称为定理结构。LaTeX 使用下面的命令支持此类结构的输入与排版：

\newtheorem{*name*}[*counter*]{*text*}[*section*]

注意这条命令必须放在文档的导言区中，即必须在\begin{document}命令之前出现。其中必选参数 name 是用来标识此定理结构的关键字，必选参数 text 代表最终将在文档中打印出来的真正的定理名称。两个可选参数 counter 和 section 都是用来确定定理结构的编号的：counter 参数指明该定理结构前驱定理的名称，这样新的定理将接着它的前驱定理进行编号；section 参数指定一个分节单元，新的定理将在该单元中进行编号。

文档导言区中使用了\newtheorem 命令之后，就可以在文档中使用形如下面的命令结构定义自己的定理结构了：

```
\begin{name}[text]
    This is my interesting theorem
\end{name}
```

其中可选参数 text 给出定理使用的诸如标题之类的文本。下面是一个例子，注意 Jery 定理使用 text 可选参数后的结果

Law 1 *Don't hide in the witness box*

Jury 2 (The Twelve) *It could be you! So beware and see law 1*

Law 3 *No, No, No*

```
% definitions for the document
% preamble
\newtheorem{law}{Law}
\newtheorem{jury}[law]{Jury}
%in the document
\begin{law} \label{law:box}
    Don't hide in the witness box
\end{law}
\begin{jury}[The Twelve]
It could be you! So beware and
see law \ref{law:box}\end{jury}
\begin{law}No, No, No\end{law}
```

上面例子中的 Jury 定理与 Law 定理使用了相同的计数器，因此它的编号同别的 Law

定理一起按顺序排列，再看下面的例子：

Murphy 4.10.1 *What can go wrong, will go wrong.*

```
\newtheorem{mur}{Murphy}[section]
\begin{mur} What can go wrong.
will go wrong. \end{mur}
```

注意：这里 Murphy 定理使用了当前节中的定理编号计数器。当然，你也可以指定使用别的分节单元（如别的章次或某个小节等）的定理计数器。

第五章 AMS 扩充的数学控制

基本LaTeX已经提供了一些数学排版功能，正如前面一章所述。如果需要重复输入复杂的公式或其他数学结构，定义新的命令或环境以便减轻打字的负担则是我们这些用户的要求。美国数学协会(AMS)考虑到这种情况并对基本LaTeX进行了扩充，定义了AMS-TeX标准，最近这一标准以一组包的形式（统称为AMS-LaTeX包）移植到了LaTeX标准当中。由于AMS-TeX中有时必须处理数学字体问题，所以相应地定义了一些称为AMS字体包的包。这些字体包不随AMS-LaTeX包一块而是单独发布。本章将介绍AMS-TeX所做的数学排版扩充的主要内容

5.1 公式中的字体和符号

5.1.1 数学符号

4.3节中我们介绍了标准LaTeX所定义的数学符号，AMS-TeX扩充了许多AMS字体的数学符号，这些符号的外观及其输入命令分别如表5.1到表5.7所示。当我们指明使用amssymb包后，这些附加的数学符号就可以使用了。如果你不需要安装所有的附加符号而只希望安装其中的一部分，可以包含amsfonts包并使用其中的\DeclareMathSymbol命令进行筛选

表 5.1 分隔定界符

⌜	\ulcorner	⌝	\urcorner	⌞	\llcorner	⌟	\lrcorner

表 5.2 希腊语和希伯来语字母

F	\digamma	$\varkappa$	\varkappa	ℶ	\beth	ℸ	\daleth	ℷ	\gimel

表 5.3 箭 头 符 号

⇠	\dashleftarrow	⇢	\dashrightarrow	⊸	\multimap	⇇	\leftleftarrows
⇉	\rightrightarrows	⇈	\upuparrows	⇆	\leftrightarrows	⇄	\rightleftarrows
⇊	\downdownarrows	⇐	\Lleftarrow	⇒	\Rrightarrow	↿	\upharpoonleft
↞	\twoheadleftarrow	↠	\twoheadrightarrow	↾	\upharpoonright	↢	\leftarrowtail
↣	\rightarrowtail	⇂	\downharpoonleft	⇋	\leftrightharpoons	⇌	\rightleftharpoons
⇃	\downharpoonright	↰	\Lsh	↱	\Rsh	⇝	\rightsquigarrow
↫	\looparrowleft	↬	\looparrowright	↭	\leftrightsquigarrow	↶	\curvearrowleft
↷	\curvearrowright	↺	\circlearrowleft	↻	\circlearrowright		

表 5.4 二元关系符号

$\lessdot$	\lessdot	$\gtrdot$	\gtrdot	$\doteqdot$	\doteqdot, \Doteq	$\leqslant$	\leqslant
$\geqslant$	\geqslant	$\risingdotseq$	\risingdotseq	$\eqslantless$	\eqslantless	$\eqslantgtr$	\eqslantgtr
$\fallingdotseq$	\fallingdotseq	$\leqq$	\leqq	$\geqq$	\geqq	$\eqcirc$	\eqcirc
$\lll$	\lll, \llless	$\ggg$	\ggg, \gggtr	$\circeq$	\circeq	$\lesssim$	\lesssim
$\gtrsim$	\gtrsim	$\triangleq$	\triangleq	$\lessapprox$	\lessapprox	$\gtrapprox$	\gtrapprox
$\bumpeq$	\bumpeq	$\lessgtr$	\lessgtr	$\gtrless$	Gtrless	$\Bumpeq$	\Bumpeq
$\lesseqgtr$	\lesseqgtr	$\gtreqless$	\gtreqless	$\thicksim$	\thicksim	$\lesseqqgtr$	\lesseqqgtr
$\gtreqqless$	\gtreqqless	$\thickapprox$	\thickapprox	$\preccurlyeq$	\preccurlyeq	$\succcurlyeq$	succcurlyeq
$\approxeq$	\approxeq	$\curlyeqprec$	\curlyeqprec	$\curlyeqsucc$	\curlyeqsucc	$\backsim$	\backsim
$\precsim$	\precsim	$\succsim$	\succsim	$\backsimeq$	\backsimeq	$\precapprox$	\precapprox
$\succapprox$	\succapprox	$\vDash$	\vDash	$\subseteqq$	\subseteqq	$\supseteqq$	\supseteqq
$\Vdash$	\Vdash	$\Subset$	\Subset	$\Supset$	\Supset	$\Vvdash$	\Vvdash
$\sqsubset$	\sqsubset	$\sqsupset$	\sqsupset	$\backepsilon$	\backepsilon	$\therefore$	\therefore
$\because$	\because	$\varpropto$	\varpropto	$\shortmid$	\shortmid	$\shortparallel$	\shortparallel
$\between$	\between	$\smallsmile$	\smallsmile	$\smallfrown$	\smallfrown	$\pitchfork$	\pitchfork
$\vartriangleleft$	\vartriangleleft	$\vartriangleright$	\vartriangleright	$\blacktriangleleft$	\blacktriangleleft	$\trianglelefteq$	\trianglelefteq
$\trianglerighteq$	\trianglerighteq	$\blacktriangleright$	\blacktriangleright				

表 5.5 否定性二元关系和箭头符号

$\nless$	\nless	$\ngtr$	\ngtr	$\varsubsetneqq$	\varsubsetneqq	$\lneq$	\lneq
$\gneq$	\gneq	$\varsupsetneqq$	\varsupsetneqq	$\nleq$	\nleq	$\ngeq$	\ngeq
$\nsubseteqq$	\nsubseteqq	$\nleqslant$	\nleqslant	$\ngeqslant$	\ngeqslant	$\nsupseteqq$	\nsupseteqq
$\lneqq$	\lneqq	$\gneqq$	\gneqq	$\nmid$	\nmid	$\lvertneqq$	\lvertneqq
$\gvertneqq$	\gvertneqq	$\nparallel$	\nparallel	$\nleqq$	\nleqq	$\ngeqq$	\ngeqq
$\nshortmid$	\nshortmid	$\lnsim$	\lnsim	$\gnsim$	\gnsim	$\nshortparallel$	\nshortparallel
$\lnapprox$	\lnapprox	$\gnapprox$	\gnapprox	$\nsim$	\nsim	$\nprec$	\nprec
$\nsucc$	\nsucc	$\ncong$	\ncong	$\npreceq$	\npreceq	$\nsucceq$	\nsucceq
$\nvdash$	\nvdash	$\precneqq$	\precneqq	$\succneqq$	\succneqq	$\nvDash$	\nvDash
$\precnsim$	\precnsim	$\succnsim$	\succnsim	$\nVdash$	\nVdash	$\precnapprox$	\precnapprox
$\succnapprox$	\succnapprox	$\nVDash$	\nVDash	$\subsetneq$	\subsetneq	$\supsetneq$	\supsetneq
$\ntriangleleft$	\ntriangleleft	$\varsubsetneq$	\varsubsetneq	$\varsupsetneq$	\varsupsetneq	$\ntriangleright$	\ntriangleright
$\nsubseteq$	\nsubseteq	$\nsupseteq$	\nsupseteq	$\ntrianglelefteq$	\ntrianglelefteq	$\subsetneqq$	\subsetneqq
$\supsetneqq$	\supsetneqq	$\ntrianglerighteq$	\ntrianglerighteq	$\nleftarrow$	\nleftarrow	$\nrightarrow$	\nrightarrow
$\nleftrightarrow$	\nleftrightarrow	$\nLeftarrow$	\nLeftarrow	$\nRightarrow$	\nRightarrow	$\nLeftrightarrow$	\nLeftrightarrow

表 5.6 二元运算符号

$\dotplus$	\dotplus	$\centerdot$	\centerdot	$\intercal$	\intercal	$\ltimes$	\ltimes
$\rtimes$	\rtimes	$\divideontimes$	\divideontimes	$\Cup$	\Cup, \doublecup	$\Cap$	\Cap, \doublecap
$\smallsetminus$	\smallsetminus	$\veebar$	\veebar	$\barwedge$	\barwedge	$\doublebarwedge$	\doublebarwedge
$\boxplus$	\boxplus	$\boxminus$	\boxminus	$\circleddash$	\circleddash	$\boxtimes$	\boxtimes
$\boxdot$	\boxdot	$\circledcirc$	\circledcirc	$\leftthreetimes$	\leftthreetimes	$\rightthreetimes$	\rightthreetimes
$\circledast$	\circledast	$\curlyvee$	\curlyvee	$\curlywedge$	\curlywedge		

表 5.7 AMS 杂类符号

$\hbar$	\hbar	$\hslash$	\hslash	$\Bbbk$	\Bbbk	$\square$	\square
$\blacksquare$	\blacksquare	$\circledS$	\circledS	$\vartriangle$	\vartriangle	$\blacktriangle$	\blacktriangle
$\complement$	\complement	$\triangledown$	\triangledown	$\blacktriangledown$	\blacktriangledown	$\Game$	\Game
$\lozenge$	\lozenge	$\blacklozenge$	\blacklozenge	$\bigstar$	\bigstar	$\angle$	\angle
$\measuredangle$	\measuredangle	$\sphericalangle$	\sphericalangle	$\diagup$	\diagup	$\diagdown$	\diagdown
$\backprime$	\backprime	$\nexists$	\nexists	$\Finv$	\Finv	$\varnothing$	\varnothing
$\eth$	\eth	$\mho$	\mho				

5.1.2 数学字体名称命令

表5.8列出了AMS字体包所提供的数学字体命令及每种字体的样本。

表 5.8 AMS 包中可用的数学字体命令

\mathbb	产生粗体板书英文字母，如$\mathbb{NQRZ}$的排版效果为$\mathbb{NQRZ}$。（amsmath包中无此命令，需要装入amssymb包）
\mathfrak	产生Euler Fraktur字体字母，如$\mathfrak{E}=\mathfrak{mc}^2$的排版效果为$\mathfrak{E}=\mathfrak{mc}^2$，（amsmath包中无此命令，需要装入amssymb包）
\boldsymbol	用来获得粗体数字、希腊字母及其他非英文字母符号。（在amsbsy包中定义）
\pmb	产生“将就黑体”(poor man's bold)效果，如$\pmb{\oint}$排版结果为$\pmb{\oint}$，$\pmb{\triangle}$排版结果为$\pmb{\triangle}$。（在amsbsy包中定义）
\text	使用数学模式外部文本模式下的当前字体产生通常格式的文本，如$E=mc^2\quad\text{(Einstein)}$的排版结果为$E=mc^2\quad\text{(Einstein)}$。（在amstext包中定义）

在amsmath包中，\boldsymbol字体用于产生数学模式下的单个粗体数学符号、希腊字母等等除了英文字母以外的所有符号，如\boldsymbol{\infty}，\boldsymbol{+}，\boldsymbol{\pi}，\boldsymbol{0}等等；单个粗体英文字母要使用\mathbf命令产生。

对于那些因为当前可用字体中不存在相应的黑体版的数学符号来说，\boldsymbol命令将不起作用，为此AMS-TeX提供了一种“将就黑体”命令来模拟产生黑体效果，具体做

法是将同一个字符输出若干遍、每次向右移动一段微小的偏移量。不过这条命令只能用于 cmex 字体下的大操作符符号以及 msam 或 msbm 字体下的 AMS 所扩充的数学符号。

为了使一个数学公式整个变粗（黑）或者尽量完全变粗（如前所述，因受可用字体限制，有些符号可能无法简单地变粗），可以在公式的前面使用\boldmath命令。

在标准LaTeX中，控制序列\mathbf{\hat{A}}将在粗体A字母上方产生一个粗体的发音标注符号，但是\mathcal{\hat{A}}将不能产生预期的效果，因为\mathcal字体中不包括各种发音标注符号。在amsmath包中，如果遇到新字体中不包括发音标注的情况，则都从\mathrm字体中获取发音标注符号。除了\mathcal字体外，\mathbb字体和\mathfrak字体也不包括发音标注符号。

5.2 组合符号、定界符和运算符

本节介绍amsmath包中在组合符号和大号定界符等方面对LaTeX进行的命令扩充。所举的例子中使用了amsmath包中定义的有关对齐方式的若干环境，有关这些环境的详细介绍请见后文中的5.4节。

5.2.1 多重积分符号

\iint, \iiint和\iiiint分别产生二重、三重和四重积分符号，每个积分符号中间的间距都作了适当调整。\idotsint产生用多个圆点(\dots)分隔的两个积分符号。如：

$$\iint_V \mu(u,v)\,du\,dv \tag{5.1}$$

$$\iiint_V \mu(u,v,w)\,du\,dv\,dw \tag{5.2}$$

$$\iiiint_V \mu(t,u,v,w)\,dt\,du\,dv\,dw \tag{5.3}$$

$$\idotsint_V \mu(u_1,\dots,u_k) \tag{5.4}$$

```
\begin{gather}
\iint\limits_V\mu(u,v)\,du\,dv\\
\iiint\limits_V\mu(u,v,w)\,du\,dv\,dw\\
\iiiint\limits_V\mu(t,u,v,w)\,dt\,du\,dv\,dw\\
\idotsint\limits_V\mu(u_1,\dots,u_k)
\end{gather}
```

5.2.2 文本上下的箭头符号

标准 LaTeX 提供了\overrightarrow 命令和\overleftarrow 命令，分别可以在文本的上方加上右向和左向的箭头修饰。AMS 扩充了类似的一些命令，如：

$$\overrightarrow{\psi_\delta(t)E_t h}=\underrightarrow{\psi_\delta(t)E_t h}$$
$$\overleftarrow{\psi_\delta(t)E_t h}=\underleftarrow{\psi_\delta(t)E_t h}$$
$$\overleftrightarrow{\psi_\delta(t)E_t h}=\underleftrightarrow{\psi_\delta(t)E_t h}$$

```
\begin{align*}
    \overrightarrow{\psi_\delta(t)E_t h}&
    =\underrightarrow{\psi_\delta(t)E_t h}\\
    \overleftarrow{\psi_\delta(t)E_t h}&
```

```
=\underleftarrow{\psi_\delta(t)E_t h}\\
                                    \overleftrightarrow{\psi_\delta(t)E_t h}&
                                    =\underleftrightarrow{\psi_\delta(t)E_t h}\\
                                \end{align*}
```

用于上下标文字时，系统也可以为这些箭头确定合适的大小，例如

$\int_{\overrightarrow{uv}} vt\,dt$ \int_{\overrightarrow{uv}} vt\,dt $

5.2.3 多个圆点符号

省略号几乎总是要使用\dots命令产生，对于标准LaTeX来说，其放置位置（放在一行的基线上还是行的垂直中央）由系统根据后面的内容自动确定：如果后面一个字符是加号或其他二元运算符，则放在垂直中央位置；如果后面一个字符是逗号，则放在基线上。这些缺省设置在amsmath包中可以改变，以适应用户不同的习惯。

如果省略号碰巧落在数学公式或数学环境的末尾，则其后将为\end, \)或$之类的内容，使得系统无法知道将省略号放在何处，此时我们需要使用下面这些命令显式地告诉系统：\dotsc命令的意思是当作后跟逗号来处理；\dotsb命令的意思是当作后跟二元运算或关系符来处理；\dotsm的意思是当作多个圆点乘号来处理；\dotsi命令的意思是当作后跟积分符号来处理（放在与积分号垂直中央位置平齐处）。省略号左右空出的距离由系统自动调节。例如：

A series $H_1,H_2\dots$, a regional sum $H_1+H_2+\cdots$, an orthogonal product $H_1H_2\cdots$, and an infinite integral $\int_{H_1}\int_{H_2}\cdots$

```
A series $H_1,H_2,\dotsc$,
a regional sum $H_1+H_2+\dotsb$, an
orthogonal product $H_1H_2\dotsm$, and
an infinite integral \[\int_{H_1}\int_{H_2}\dotsi\].
```

5.2.4 双重发音标注

如果我们想在字母 A 的头上叠放两个发音标注（比如说两个锐音标注），在标准 LaTeX 中可以使用\acute{\acute{A}}命令组合，如：

$\acute{\acute{A}}$ $\bar{\bar{B}}$ $\breve{\breve{C}}$ $\grave{\grave{D}}$
$\ddot{\ddot{E}}$ $\dot{\dot{F}}$ $\check{\check{G}}$ $\hat{\hat{H}}$
$\tilde{\tilde{I}}$ $\vec{\vec{J}}$

```
\begin{gather*}
\Acute{\Acute{A}} \qquad\Bar{\Bar{B}} \qquad
\Breve{\Breve{C}} \qquad\Check{\Check{D}} \\
\Ddot{\Ddot{E}} \qquad\Dot{\Dot{F}} \qquad
\Grave{\Grave{G}} \qquad\Hat{\Hat{H}} \\
\Tilde{\Tilde{I}} \qquad\Vec{\Vec{J}}
\end{gather*}
```

这种产生双重叠加发音标注的操作既复杂又会使 LaTeX 文件的处理速度下降。如果你经常需要这种操作，可以在文档中装入 amsxtra 包，该包中定义了可在文档导言区中使用的\accentedsymbol 命令，它可以把前述命令组合所产生的双重发音标注放进一个文本盒子中，以便加快文档的处理速度。\accentedsymbol 命令的使用与\newcommand 命令有些相似：

This is a double hat $\Ahathat{A}$ and this $\dbardot{\delta}$ a delta with a bar and a dot.

```
\accentedsymbol{\Ahathat}{\Hat{\Hat A}}
\accentedsymbol{\dbardot}{\Dot{\Bar \delta}}
```

```
This is a double hat \(\Ahathat\) and this
    \(\dbardot\) a delta with a bar and a dot.
```

5.2.5 宽发音标注

有些发音标注具有较宽的形式，如$\widehat {xy},\widetilde{xy}$将分别产生宽的抑扬标注和宽的波浪标注。因为这些宽的发音标注都有一定的最大宽度限制，另外较宽时形状也不大美观，为此 amsxtra 包引入了不同的符号来处理过宽的发音标注：用(AmBD)^格式来代替原来的 $\widehat{AmBD}$ 格式。Amsxtra 包提供了下面这些控制序列来轻松产生这类新的注音符号：

(AmBD)_ (AmBD)\^ (5.5)
(AmBD)' (AmBD)` (5.6)
(AmBD)'' (AmBD)''' (5.7)
(AmBD)˘ (5.8)

```
\begin{gather}
(AmBD)\sphat \qquad (AmBD)\spcheck \\
(AmBD)\sptilde \qquad (AmBD)\spdot \\
(AmBD)\spddot \qquad(AmBD)\spdddot \\
(AmBD)\spbreve
\end{gather}
```

5.2.6 圆点发音标注

标准 LaTeX 中提供了\dot 和\ddot 命令，分别在字母上产生一个圆点和两个圆点的发音标注。AMS-TeX 另外增加了三个圆点和四个圆点的发音标注命令，分别为\dddot 和\ddddot 命令。

5.2.7 开方符号

标准 LaTeX 在排版开方符号时，有时候开方的次（指）数的位置安排得不是很好。使用 amsmath 提供的\leftroot{offset}和\uproot{offset}命令可以调整开方次数的位置，其中参数 offset 为调整的位移，offset 为正时开方次数分别向左和上移动，offset 为负时开方次数分别向右和下移动。位移的长度单位相当小，这样才能保证调整的效果，因为开方次数允许移动的区域很小。下面的例子中开方次数β一次被左移了 2 个长度单位，一次被上移了 4 个长度单位：

$\sqrt[\beta]{k}$ $\sqrt[\leftroot{2}\uproot{4}\beta]{k}$

```
\[ \sqrt[\beta]{k} \qquad
\sqrt[\leftroot{2}\uproot{4}\beta]{k} \]
```

5.2.8 带边框公式

\boxed 命令同\fbox 命令相似，在其参数的四周放置矩形边框，所不同的是其参数是处于数学模式而不是处于文本模式。例如：

$$\boxed{W_t - F \subseteq V(P_i) \subseteq W_t}$$

```
\[ \boxed{W_t - F \subseteq
V(P_i) \subseteq W_t} \]
```

5.2.9 可伸缩箭头

AMS-TeX 定义的\xleftarrow 命令和\xrightarrow 命令，可以产生左箭头或右箭头，这些可以自动伸缩，以与其上下特别长的文本相适应。箭头上面的文本是必选参数，面箭头下面的文本是可选参数。如：

$$0\xleftarrow[\zeta]{\alpha} F\times\triangle[n-1]\xrightarrow{\partial_0\alpha(b)}E^{\partial_0 b}$$

```
\[0 \xleftarrow[\zeta]{\alpha}
F\times\triangle[n-1]
\xrightarrow{\partial_0\alpha(b)}
E^{\partial_0b}\]
```

5.2.10 \overset, \underset 和\sideset 命令

标准 LaTeX 提供的\stackrel 命令可以将文本放置在分隔定界符或二元关系符的上面。amsmath 引入了几个更为通用的命令，\overset{syb1}{syb2}命令和\underset{syb1}{syb2}命令分别将任意符号 syb1 放在任意符号 syb2 的上面或下面，如：

$$\overset{*}{X}\quad\underset{*}{X}\quad\overset{a}{\underset{b}{X}}$$

```
\[ \overset{*}{X} \qquad
\underset{*}{X} \qquad
\overset{a}{\underset{b}{X}} \]
```

假设我们想在求和符号的右上角放置一个撇号，如果不带求和上下界，则可以使用标准 LaTeX 提供的\nolimits 命令，例如：

$$\sum\nolimits' E_n \tag{5.9}$$

```
\begin{equation}
    \sum\nolimits' E_n.
\end{equation}
```

但是如果求和符号带有求和上下界，而我们又想放置上下标的话，问题就比较复杂了，也许下面的例子提供了一个思路，即将上标当作求和符号正上方内容的一部分，将下标当作求和符号正下方内容的一部分，尽管上下标显示的位置既受到限制又需要我们仔细地调整：

$$\sum_{n<k,\;n\ \mathrm{odd}}^{nE_n} \tag{5.10}$$

```
\begin{equation}
    \sum_{n<k,\;n\ \mathrm{odd}}nE_n
\end{equation}
```

amsmath 定义的\sideset{left}{right}命令可以很方便地解决上述问题，该命令在大型运算符（如求和运算 $\sum$ 和求积运算 $\prod$）左右两边的上下标位置放置文本或符号，其中 left 参数代表放在运算符左边的上下标内容，right 代表放在运算符右边的上下标内容。如：

$$\sideset{_1^2}{_3^4}\prod_k \qquad \sideset{}{'}\sum_{0\le i\le m} E_i\beta x$$

```
\sideset{_1^2}{_3^4}\prod_k \qquad
\sideset{}{'}\sum_{0\le i\le m} E_i\beta x\]
```

5.2.11 \smash 命令

普通 TeX 中的\smash{text}命令保持参数文本 text 的内容，但是将顶部和底部压缩，使顶部低于当前行上标位置的底部，同时使底部高于当前行下标位置的顶部。Amsmath 包给

该条命令引入了一个可选参数，变成\smash[where]{text}格式，where 取 t 时只压缩 text 顶部，where 取 b 时只压缩 text 底部，例如：

$$X_j=(1/\sqrt{\smash[b]{\lambda_j}})X_j' \qquad X_j=(1/\sqrt{\lambda_j})X_j'$$

```
\[X_j=(1/\sqrt{\smash[b]{\lambda_j}})X_j'
\qquad X_j=(1/\sqrt{\lambda_j})X_j' \]
```

从上面的例子中我们可以看出，左边公式中的开方符号的左下角要比不用\smash 命令时（右边公式）高出一个下标的高度。

5.2.12 \text 命令

\text 命令在 amstext 包中定义，主要作用同标准 LaTeX 的\mbox 命令相似，如传统的控制序列_{\mbox{scriptsize word or phrase}}产生 $_{\text{word or phrase}}$ 的效果，也可以用_{\text{word or phrase}}控制序列来实现，不过后者除了更加直观外输入更加容易，因为\text 命令的参数文本的字号已经由系统自动确定了。例如：

$$\mathbf{y}=\mathbf{y}' \quad \text{if and only if} \quad y'_k=\delta_k y_{\tau(k)}$$

```
\[ \mathbf{y}=\mathbf{y}' \quad
\text{if and only if} \quad
y'_k=\delta_k y_{\tau(k)} \]
```

5.2.13 数学运算（函数）名称

在数学模式下，诸如 log, sin, lim 等数学函数习惯上都使用 roman 字体，以免同使用数学斜体的数学变量名相混淆，为此 LaTeX 对比较常用的数学函数名进行了预定义，如\log, \sin, \lim 等等（详见 4.4.2 节）。但是新的数学函数在不断出现，为此 amsmath 包提供了两条命令以声明新的函数名：

```
\DeclareMathOperator{\fname}{typo}
\DeclareMathOperator*{\fname}{typo}
```

其中 fname 为新定义的函数名，typo 为该函数名的排版格式。带星号的命令与不带星号的命令的惟一差别在于对函数名所带的上下标的处理，例如控制序列

```
\DeclareMathOperator*{\esssup}{ess\,sup}
\DeclareMathOperator{\meas}{meas}
\newcommand{\abs}[1]{\lvert#1\rvert}
\newcommand{\norm}[1]{\lVert#1\rVert}
\begin{align*}
\norm{f}_\infty &=
\esssup_{x\in R^n}\abs{f(x)} \\
\meas_1\{u\in R_+^1\colon f^*(u)>\alpha\} &=
\meas_n\{x\in R^n\colon\abs{f(x)}\geq\alpha\}\qquad\forall\alpha>0.
\end{align*}
```

将产生如下的结果：

$$\|f\|_\infty = \operatorname*{ess\,sup}_{x\in R}|f(x)|$$

$$meas_1\{u\in R_+^1 : f^*(u)>\alpha\} = meas_n\{x\in R^n : |f(x)|\geq\alpha\} \qquad \forall\alpha>0$$

在 amsmath 包中，\varlimsup, \varliminf, \varinjlim 和\varprojlim 等几个数学运算是预先定义好的，下面的例子说明了它们的应用：

```
\begin{gather}
\varlimsup_{n\rightarrow\infty}
\mathcal{Q}(u_n,u_n-u^{\#}) \le 0 \\
\varliminf_{n\rightarrow\infty}
\left|a_{n+1}\right|/\left|a_n\right| = 0 \\
\varinjlim (m_i^\lambda\cdot)^* \le 0 \\
\varprojlim_{p\in S(A)}A_p \le 0
\end{gather}
```

产生的结果为

$$\overline{\lim}_{n\to\infty} Q(u_n, u_n - u^*) \le 0 \tag{5.11}$$

$$\underline{\lim}_{n\to\infty} |a_{n+1}|/|a_n| = 0 \tag{5.12}$$

$$\underline{\lim}(m_j^{\lambda}\bullet)^* \le 0 \tag{5.13}$$

$$\underline{\lim}_{p\in S(A)} A_p \le 0 \tag{5.14}$$

为了更灵活地使用各种竖线符号，amsmath 定义了\lvert, \rvert, \lVert 和\rVert 等几个新的命令，读者可以将它们同 LaTeX 的\langle 和\rangle 命令进行比较。

5.2.14 \mod 及其相关命令

amsopn 包提供了\mod, \bmod, \pmod 以及\pod 命令来处理求模函数(mod)周围的留空问题，其中\bmod 和\pmod 命令在标准 LaTeX 中也有定义，但是 amsopn 对\pmod 所留空白的大小进行了调整。\mod 和\pod 命令是\pmod 命令的变种，前者省去了括号，后者省去了 mod 字样而留下了括号。例如：

$$\gcd(k, l \bmod k) \tag{5.15}$$

```
\begin{equation}
\gcd (k , l \bmod k)
\end{equation}
```

$$u \bullet v + 1 \pmod{n^2} \tag{5.16}$$

$$u \bullet v + 1 \mod{n^2} \tag{5.17}$$

$$u \bullet v + 1 \pod{n^2} \tag{5.18}$$

```
\begin{align}
u & \equiv v + 1 \pmod{n^2} \\
u & \equiv v + 1 \mod{n^2} \\
u & \equiv v + 1 \pod{n^2}
\end{align}
```

5.2.15 分式及其相关结构

标准 LaTeX 提供了\frac 命令处理分式结构，amsmath 另外提供了\dfrac 命令作为控制序列\displaystyle\frac 的缩写命令，提供了\tfrac 命令作为控制序列\textstyle\frac 的缩写命令：

$$\frac{1}{k}\log_2 c(f) \quad \tfrac{1}{k}\log_2 c(f)$$

and

$$\sqrt{\frac{1}{k}\log_2 c(f)} \quad \sqrt{\tfrac{1}{k}\log_2 c(f)}$$

```
\[ \frac{1}{k}\log_2 c(f)\quad
\tfrac{1}{k}\log_2 c(f) \]
and
$ \sqrt{\frac{1}{k}\log_2 c(f)}\quad
\sqrt{\tfrac{1}{k}\log_2 c(f)} $.
```

对于组合二项式表达式，amsmath 包定义了\binom, \dbinom 和\tbinom 命令：

$$\binom{k}{1}2^{k-1}+\binom{k}{2}2^{k-2} \qquad (5.19)$$

and $\binom{k}{1}2^{k-1}+\binom{k}{2}2^{k-2}$

```
\begin{equation}
\binom{k}{1}2^{k-1}+
\tbinom{k}{2}2^{k-2}
\end{equation}
and $\binom{k}{1}2^{k-1}+
\dbinom{k}{2}2^{k-2}$.
```

\binom 命令和它的变种\dbinom 与\tbinom，以及\frac 命令和它的变种\dfrac 与\tfrac，都是用一个带 6 个参数的一般形式的分式结构命令\genfrac 实现的：

\genfrac{*ldelim*}{*rdelim*}{*thick*}{*style*}{*num*}{*denom*}

其中参数 ldelim 和 rdelim 分别表示左右分界符；参数 thick 用来定义分数线的宽度（如\binom 命令定义时将该参数设为 0），缺省时设为"normal"；参数 style 设置数学环境类型，取值为 0 到 3 之间的整数，分别表示\displaystyle, \textstyle, \scriptstyle 以及\scriptscriptstyle；参数 num 和 demon 分别表示分子和分母文本。下面用\frac, \tfrac 以及\binom 命令的具体定义来说明\genfrac 命令的用法：

```
\newcommand{\frac}[2]{\genfrac{}{}{}{}{#1}{#2}}
\newcommand{\tfrac}[2]{\genfrac{}{}{}{1}{#1}{#2}}
\newcommand{\binom}[2]{\genfrac{(}{)}{0pt}{}{#1}{#2}}
```

5.2.16 连续分式结构

连续分式结构可以使用\cfrac[lIr]{num}{denom}命令产生，其中 num 和 denom 参数分别表示分子和分母文本，可选参数表示分子的位置（l 表示放在左边，r 表示放在右边，确省时表示居中）。例如：

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{3}+\cfrac{1}{\sqrt{4}+\cfrac{1}{\sqrt{5}+\cfrac{1}{\sqrt{6}+\cdots}}}}} \qquad (5.20)$$

```
\begin{equation}
\cfrac{1}{\sqrt{2}+
\cfrac{1}{\sqrt{3}+
\cfrac{1}{\sqrt{4}+
\cfrac[l]{1}{\sqrt{5}+
\cfrac[r]{1}{\sqrt{6}+\dotsb
}}}}}
\end{equation}
```

5.2.17 超大定界符

为了更好地控制数学定界符的尺寸，基本 TeX 引入了\big, \Big, bigg 和\Bigg 这 4 个命

令，每条命令的参数是要产生定界符，定界符尺寸依次递增。所有可以跟在\left 或\right 命令后面的定界符（见表 4.8、表 4.9 和表 5.1）都可以用这些命令产生。另外这 4 个命令还各有 3 种变体：开始（左）符号（如\bigl），二元关系符号（如\Bigm），结束（右）符号（如\Biggr） 然而，在基本 TeX 中这些命令产生的定界符其大小都是固定的，但是在 amsmath 包中定界符的大小与其周围的文档内容自动相协调。例如：

$$\left(E_y\int_0^{t_\varepsilon} L_x, y^x(s)\varphi(x)ds\right)$$

$$\left(E_y\int_0^{t_\varepsilon} L_x, y^x(s)\varphi(x)ds\right)$$

```
\biggl(\mathbf{E}_{y}\int_0^{t_\varepsilon}
L_{x,y^x(s)}\varphi(x)\,ds \biggr)
\]
{\Large
\[
\biggl(\mathbf{E}_{y}\int_0^{t_\varepsilon}
L_{x,y^x(s)}\varphi(x)\,ds \biggr)
\]}
```

5.3 矩阵式环境和换向图

5.3.1 cases 环境

类似于分段函数的 case 结构，可以使用 cases 环境来建立。如：

$$P_{r-j}=\begin{cases}0 & \text{if } r-j \text{ is odd},\\ r!(-1)^{(r-j)/2} & \text{if } r-j \text{ is even}.\end{cases} \tag{5.21}$$

```
\begin{equation}
P_{r-j}=
\begin{cases}
0& \text{if $r-j$ is odd},\\
r!\,(-1)^{(r-j)/2}& \text{if $r-j$ is even}.
\end{cases}
\end{equation}
```

注意上面例子中\text 命令以及内置数学环境的应用。

5.3.2 matrix（矩阵）类环境

矩阵类环境类似于标准 LaTeX 中的 array 环境，但是它们不需要用参数来指明列的格式，而是提供了一种缺省的格式：最多到 10 列，每列居中对齐。下面的例子说明如何使用 matrix, pmatrix, bmatrix, vmatrix 以及 Vmatrix 这几种矩阵类环境。

$$\begin{matrix}0 & 1\\ 1 & 0\end{matrix}\quad \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\quad \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$$

$$\begin{vmatrix}a & b\\ c & d\end{vmatrix}\quad \begin{Vmatrix}1 & 0\\ 0 & 1\end{Vmatrix}$$

```
\begin{gather*}
\begin{matrix} 0 & 1 \\ 1 & 0\end{matrix} \quad
\begin{pmatrix}0 &-i \\ i & 0\end{pmatrix}\quad
\begin{bmatrix}1 & 0 \\ 0 &-1\end{bmatrix}\\
\begin{vmatrix}a & b \\ c & d\end{vmatrix}\quad
\begin{Vmatrix}1 & 0 \\ 0 & 1\end{Vmatrix}
\end{gather*}
```

最大列数由计数器 MaxMatrixCols 确定，我们可以用标准 LaTeX 的计数器命令改变它的值。比如假设要产生包含 19 或 20 列的大矩阵，我们可以使用诸如下面的控制序列：

```
\begin{equation}
\setcounter{MaxMatrixCols}{20}
A=\begin{pmatrix}
...&...&...&...&...&...&...&...&...&...&...&...& ... \\
... \\
...
\end{pmatrix}
\end{equation}
\setcounter{MaxMatrixCols}{10}
```

由于计数器对于 LaTeX 系统来说是全局性的，因此在输出大矩阵以后一般要把 MaxMatrixCols 恢复其缺省值 10，因为如果它的值较大，即使输出小矩阵 LaTeX 也要花费更大的力气。

AMS-TeX 还提供了一种 smallmatrix 环境，专门用来产生小尺寸的矩阵。例如：

```
To show the effect of the matrix on surrounding
 lines inside a paragraph, we put it here:
\begin{math}
    \left( \begin{smallmatrix}
    a&b\\ c&d
    \end{smallmatrix} \right)
\end{math}
and follow it with enough text to ensure that
there is at least one full line below the matrix.
```

To show the effect of the matrix on surrounding lines inside a paragraph, we put it here: $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right)$ and follow it with enough text to ensure that there is at least one full line below the matrix.

在矩阵环境中，如果要得到贯穿指定列数的一串圆点，可以使用下面命令：

```
\hdotsfor[spacing-factor]{number}
```

其中可选参数 spacing-factor 用来改变圆点之间的间距（比例因子，缺省情况为 1）；必选参数 number 表示要贯穿的列数。例如控制序列

```
\[ W(\Phi)= \begin{Vmatrix}
\dfrac\varphi{(\varphi_1,\varepsilon_1)} &0&
\hdotsfor{2} &0 \\
\dfrac{\varphi k_{n2}}{(\varphi_2,\varepsilon_1)} &
\dfrac\varphi{(\varphi_2,\varepsilon_2)} &0&\dots&0\\
\hdotsfor{5} \\
\dfrac{\varphi k_{n1}}{(\varphi_n,\varepsilon_1)} &
\dfrac{\varphi k_{n2}}{(\varphi_n,\varepsilon_2)}&\dots&
\dfrac{\varphi k_{n\,(n-1)}}{(\varphi_n,\varepsilon_{n-1})}&
\dfrac{\varphi}{(\varphi_n,\varepsilon_n)}
\end{Vmatrix}\]
```

产生的结果为

$$W(\Phi)=\left\|\begin{array}{cccc}\dfrac{\varphi}{(\varphi_1,\varepsilon_1)} & 0 & \cdots\cdots & 0\\ \dfrac{\varphi k_{n2}}{(\varphi_2,\varepsilon_1)} & \dfrac{\varphi}{(\varphi_2,\varepsilon_2)} & 0 \quad \cdots & 0\\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots\\ \dfrac{\varphi k_{n1}}{(\varphi_n,\varepsilon_1)} & \dfrac{\varphi k_{n2}}{(\varphi_n,\varepsilon_2)} & \cdots \quad \dfrac{\varphi k_{n(n-1)}}{(\varphi_n,\varepsilon_{n-1})} & \dfrac{\varphi}{(\varphi_n,\varepsilon_n)}\end{array}\right\|$$

5.3.3 \substack 命令

\substack 命令可以用来排版多行的居中对齐的上标或下标文本，文本中使用\\命令换行，只要一般上下标能够出现的地方，都可以出现这条命令。如：

$$\sum_{\substack{0\le i\le m\\ 0<j<n}} P(i,j) \qquad (5.22)$$

```
\begin{equation}
\sum
_{\substack{0\le i\le m\\ 0<j<n}}
P(i,j)
\end{equation}
```

如果多行上下标想左对齐，可以使用 subarray 环境来代替\substack 命令：

$$\sum_{\substack{i\in A\\ 0<j<n}} P(i,j) \qquad (5.23)$$

```
\begin{equation}
\sum_{\begin{subarray}{l}
i\in \Lambda\\ 0<j<n
\end{subarray}}
P(i,j)
\end{equation}
```

5.3.4 换向图

AMS-TeX 中的换向图命令不是包含在 amsmath 包中，而是以单独的 amscd 包发布，主要目的是为了节约不用换向图功能的用户的机器内存开销。对于较为复杂的换向图来说需要使用图形环境，但对于不含有对角线方向的斜箭头的简单图，amscd 包中的 CD 环境和相应的命令更方便简单一些。例如：

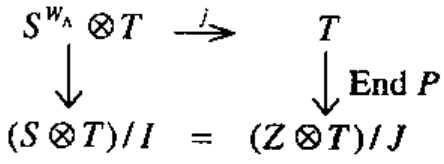
$$\begin{array}{ccc} S^{W_\Lambda}\otimes T & \xrightarrow{j} & T \\ \downarrow & & \downarrow \text{End } P \\ (S\otimes T)/I & = & (Z\otimes T)/J \end{array}$$

```
\DeclareMathOperator{\End}{End}
\[\begin{CD}
S^{\mathcal{W}_\Lambda}\otimes T
@>j>> T \\
@VVV @VV{\End P}V\\
(S\otimes T)/I @= (Z\otimes T)/J
\end{CD}\]
```

CD 环境中，命令@>>>, @<<<, @VVV 和@AAA 分别产生向右、向左、向下和向上的箭头，如果键盘上没有尖括号，也可以用同方向的圆括号代替。对于水平箭头，第一个>（或<）与第二个>（或<）之间的内容将放在箭头的上方；第二个>（或<）与第三个>

（或<）之间的内容将放在箭头的下方。类似地，对于垂直箭头，第一个 A（或 V）与第二个 A（或 V）之间的内容将放在箭头的左边；第二个 A（或 V）与第三个 A（或 V）之间的内容将放在箭头的右边。

使用下面的标准 LaTeX 控制序列也可以得到相似的结果，尽管效果可能要差一些，因为使用 amscd 包时，得到的水平箭头更长一些，图中各个元素之间的空格控制也得到了改进：



```
\[\begin{array}{ccc}
S^{\mathcal{W}_\Lambda}\otimes T & \stackrel{j}{\longrightarrow} & T \\
\Big\downarrow & & \Big\downarrow\vcenter{%
\rlap{$\scriptstyle{\mathrm{End}}\,P$}} \\
(S\otimes T)/I & = & (Z\otimes T)/J
\end{array}\]
```

5.4 多行公式的对齐结构

amsmath 包定义了几个创建多行显示公式的环境，它们同标准 LaTeX 的 equation 和 eqnarray 环境功能相似。后面各节中将介绍下面这些结构，星号版结构将不带公式编号：

align, align*	在一个地方（对齐点）对齐
flalign, flalign*	使公式拉宽的 align 结构的变体
alignat, alignat*	带间距控制的对齐结构
equation, equation*	单行公式
gather, gather*	不对齐的公式组合
multline, multline*	不对齐的多行公式（只有一个公式编号）
split	对齐地分裂长公式

这些多行显示环境中有一些允许用户对齐公式的某个部分。同标准 LaTeX 提供的 eqnarray 和 eqnarray*环境相比，amsmath 包实现的结构使用了完全不同的方法标注对齐点：eqnarray 同带有{rcl}参数的 array 环境相似，使用两个&符号将要对齐的部分括起来；amsmath 定义的结构中，只需要将一个&符号放在某个字符的前面（我们称&所在位置为对齐点），该字符将与前后行中的对齐点对齐。

Amsmath 结构自动在对齐点的周围留下合适的空白，而 eqnarray 环境根据 array 参数设置的不同留下大小不等的多余空白。下面的例子中分别用 equation, align 和 eqnarray 环境排版相同的公式，从中我们可以看出各个环境的不同之处：理想情况下得到的排版结果应当完全相同，但是 eqnarray 的结果太宽了。

$$x^2+y^2=z^2 \tag{5.24}$$

```
\begin{equation}
x^2+y^2 = z^2
\end{equation}
```

$$x^2+y^2=z^2 \quad (5.25)$$
$$x^3+y^3<z^3 \quad (5.26)$$

```
\begin{align}
x^2+y^2 &= z^2 \\ x^3+y^3 &< z^3
\end{align}
```

$$x^2+y^2 = z^2 \quad (5.27)$$
$$x^3+y^3 < z^3 \quad (5.28)$$

```
\begin{eqnarray}
x^2+y^2 &=& z^2 \\ x^3+y^3 &<& z^3
\end{eqnarray}
```

5.4.1 不对齐的公式组

gather 环境用来放置两个或更多的公式，这些公式之间不进行对齐，每个公式在左右环境边界之间居中。例如：

$$(a+b)^2 = a^2 + 2ab + b^2 \quad (5.29)$$
$$(a+b)\cdot(a-b) = a^2 - b^2 \quad (5.30)$$

```
\begin{gather}
    (a + b)^2 = a^2 + 2ab + b^2\\
    (a + b) \cdot (a - b) = a^2 - b^2
\end{gather}
```

5.4.2 对齐的公式组

align 环境用来放置需要垂直对齐的两个或更多的公式（通常公式中的二元关系符号如等号需要对齐）。例如：

$$x^2 + y^2 = 1 \qquad x^3 + y^3 = 1 \quad (5.31)$$
$$x = \sqrt{1-y^2} \qquad x = \sqrt[3]{1-y^3} \quad (5.32)$$

```
\begin{align}
    x^2 + y^2 &= 1 &
    x^3 + y^3 &= 1 \\
    x &= \sqrt{1-y^2} &
    x &= \sqrt[3]{1-y^3}
\end{align}
```

使用 slign 环境建立的一个公式，其各个部分均匀地在同一行上分布，如果想控制公式中不同列之间的间隔，可以使用支持多对齐点的 alignat 环境。Alignat 环境有一个必选参数，用来指定对齐点的个数（即需要几个小的 "align"结构），如果参数的值是 n，则一行中总共要有 2n-1 个&符号，因为每个 align 结构需要一个&标注对齐点，另外两个 align 结构之间还要用一个&分隔。

flalign 是一种特殊的 align 环境，它在公式的不同部分之间填加额外的空白。例如：

$$L_1 = R_1 \qquad L_2 = R_2 \quad (5.33)$$
$$L_3 = R_3 \qquad L_4 = R_4 \quad (5.34)$$

```
\begin{align}
L_1 &= R_1 & \qquad L_2 &= R_2 \\
L_3 &= R_3 & \qquad L_4 &= R_4
\end{align}
```

$$L_1 = R_1 \quad L_2 = R_2 \quad (5.35)$$
$$L_3 = R_3 \quad L_4 = R_4 \quad (5.36)$$

```
\begin{alignat}{2}
L_1 &= R_1 & \qquad L_2 &= R_2 \\
L_3 &= R_3 & \qquad L_4 &= R_4
\end{alignat}
```

```
\begin{flalign}
L_1 = R_1 \qquad L_2 = R_2 \quad (5.37) & L_1 &= R_1 && \qquad L_2 &= R_2 \\
L_3 = R_3 \qquad L_4 = R_4 \quad (5.38) & L_3 &= R_3 && \qquad L_4 &= R_4
\end{flalign}

\begin{flalign*}
L_1 = R_1 \qquad L_2 = R_2 & L_1 &= R_1 && \qquad L_2 &= R_2 \\
L_3 = R_3 \qquad L_4 = R_4 & L_3 &= R_3 && \qquad L_4 &= R_4
\end{flalign*}
```

5.4.3 不对齐的多行公式

multline 环境是 equation 环境的变体，用来排不适合放在单行里的公式。左右两边各留下一个大小等于\multlinegap 的空白后，multline 所排公式的第一行将放在最左边，最后一行将放在最右边。\multlinegap 的值可以用 LaTeX 的\setlength 或\addtolength 命令修改。如果 multline 环境中的公式不止两行，则除了第一行和最后一行外，其余各行都在环境的显示宽度内居中，不过我们可以用\shoveleft 命令或\shoveright 命令将某一行强制到左边或右边。例如：

```
\begin{multline}
First line of equation \\
\text{First line of equation} \\
Centered Middle line \\
\text{Centered Middle line} \\
Right Middle line \\
\shoveright{\text{Right Middle line}} \\
Other centered Middle \\
\text{Other centered Middle} \\
Left Middle line \\
\shoveleft{\text{Left Middle line}} \\
Last line of equation
\text{Last line of equation}
\end{multline}
```

5.4.4 对齐的分裂长公式

同 multline 环境一样，split 环境也用于由于排版太长而一行放不下的公式。但是与 multline 环境不同的是，split 将长公式分裂得到的多行之间可以对齐，对齐点的位置也是用一个&符号标注。另外与其他 amsmath 的公式环境都不同的是，split 环境中的公式不带编号（因此 split 也没有相应的星号版），因为 split 环境一般都用于另一个公式结构（如 equation, align 或 gather 等）的内部，公式的编号将由外层的公式结构负责。例如：

```
\begin{equation}
  \begin{split}
(a+b)^4 &= (a+b)^2 (a+b)^2 \\
&= (a^2+2ab+b^2)(a^2+2ab+b^2) \\
&= a^4+4a^3b+6a^2b^2+4ab^3+b^4 \\
  \end{split}
\end{equation}
```

$$\begin{aligned}(a+b)^4 &= (a+b)^2(a+b)^2 \\ &= (a^2+2ab+b^2)(a^2+2ab+b^2) \\ &= a^4+4a^3b+6a^2b^2+4ab^3+b^4\end{aligned} \tag{5.39}$$

split 环境有一个表明公式编号放置位置的可选项（参见 5.5.5 节），其缺省情况是

centertags，表示公式的编号将放在 split 环境的垂直居中处（如上例）。假设我们指定使用 tbtags，则如果编号放在公式右边的话就放在最后一行（如下例）；如果编号放在公式左边的话就放在第一行

$$\begin{aligned}(a+b)^3 &= (a+b)(a+b)^2 \\ &= (a+b)(a^2+2ab+b^2) \\ &= a^3+3a^2b+3ab^2+b^3 \end{aligned} \tag{5.40}$$

```
\begin{equation}
\begin{split}
(a+b)^3 &= (a+b) (a+b)^2 \\
&= (a+b)(a^2+2ab+b^2) \\
&= a^3+3a^2b+3ab^2+b^3 \\
\end{split}
\end{equation}
```

5.4.5 部分显示作用的对齐环境

上一节中的 split 环境不单独使用，即不是完整的显示环境，我们称这类环境为部分显示环境，它们是可自包含的单元，可以在其他公式内部使用，也可以一个接一个地并排排列。除了 split 外，还有一些其他的部分显示公式环境：aligned, gathered 和 alignedat 。这些环境分别带有一个可选参数，用来指定相对于环境两边的文档内容和这些环境放置的垂直位置，该参数的缺省值是 c（居中），其作用如下例所示：

$$\begin{aligned} x^2+y^2 &= 1 \\ x &= \sqrt{1-y^2} \end{aligned} \qquad \begin{aligned} (a+b)^2 &= a^2+2ab+b^2 \\ (a+b)\cdot(a-b) &= a^2-b^2 \end{aligned}$$

```
\begin{equation*}
\begin{aligned}
x^2 + y^2 &= 1 \\
x &= \sqrt{1-y^2}
\end{aligned} \qquad
\begin{gathered}
(a + b)^2 = a^2 + 2ab + b^2 \\
(a + b) \cdot (a - b) = a^2 - b^2
\end{gathered}
\end{equation*}
```

同样的数学公式由于使用了不同的垂直位置可选参数而得到了完全不同的排版结果，如下例所示。注意下例中两次分别使用了[b]和[t]可选参数值。

$$\begin{aligned} x^2+y^2 &= 1 \\ x &= \sqrt{1-y^2} \end{aligned} \qquad \begin{aligned} (a+b)^2 &= a^2+2ab+b^2 \\ (a+b)\cdot(a-b) &= a^2-b^2 \end{aligned}$$

```
\begin{equation*}
\begin{aligned}[b]
x^2 + y^2 &= 1 \\
x &= \sqrt{1-y^2}
\end{aligned} \qquad
\begin{gathered}[t]
(a + b)^2 = a^2 + 2ab + b^2 \\
(a + b) \cdot (a - b) = a^2 - b^2
\end{gathered}
\end{equation*}
```

5.4.6 公式环境中的垂直间距和换页

如同在标准 LaTeX 中一样，我们也可以在所有 amsmath 公式环境中使用\\[dimension]命令在两行之间进行垂直间距控制，与 eqnarray 环境不同，amsmath 公式环境中行与行之间不允许出现空格，除非环境中使用了\displaybreak 或\allowdisplaybreaks 命令，因为这些环境中的换页可能极大地影响排版效果，应当引起文档作者的特别注意。\displaybreak 或\allowdisplaybreaks 命令必须出现在可能会引起换页的\\命令之前才能起作用。另外同标准 LaTeX 中的\pagebreak 命令一样，\displaybreak 命令也带有一个取值在 0 到 4 之间的可选参数表示允许中断的意愿的强弱：\displaybreak[0]表示“可以从这儿断开，但是不鼓励断开”；\displaybreak[4]表示强制断开（缺省值）。

\allowdisplaybreaks 命令也有一个可选参数。这条命令遵从标准 LaTeX 有关命令作用域的规则，通常将{\allowdisplaybreaks 放在需要它起作用的区域的开头，在作用域的末尾放置一个}。在\allowdisplaybreaks 命令的作用域内部，同标准 LaTeX 中一样，*命令可以用来禁止换页。

5.4.7 \intertext 命令

\intertext 命令用于在对齐结构的中间插入一两行较短的文本，其明显的优点是可以保持对齐结构的整齐性，因为如果先结束对齐结构、插入文本，再重新进入对齐结构，那么很难做到前后两个对齐结构的位置整齐。\intertext 命令只能紧跟\\或*命令后面出现。例如：

$$A_1 = N_0(\lambda;\Omega') - \phi(\lambda;\Omega'), \quad (5.41)$$

$$A_2 = \phi(\lambda;\Omega')\phi(\lambda;\Omega), \quad (5.42)$$

andfinally

$$A_3 = N(\lambda;\omega). \quad (5.43)$$

```
\begin{align}
A_1&=N_0(\lambda;\Omega') -
\phi(\lambda;\Omega'), \\
 A_2&=\phi(\lambda;\Omega')
 \phi(\lambda;\Omega), \\
\intertext{and finally}
 A_3&=\mathcal{N}(\lambda;\omega).
 \end{align}
```

此例中的文本“and finally”不属于对齐结构的显示范围并被放在紧靠页面左边注的地方。

5.5 其他杂类命令

本节中介绍 amsmath 引入的其他尚没有讨论的命令，另外给出随 AMS-LaTeX 发布的所有文档类文件的清单。

5.5.1 公式编号

除了 split 外的每一个公式环境，都有星号版和无星号版两种命令格式，其中无星号版命令产生的公式使用 LaTeX 的公式计数器自动编号，星号版命令产生的公式不带编号。通

过在\\命令之前使用\notag 命令，可以去掉每一特定行上的编号。也可以使用\tag{label}或tag*{label}命令改变缺省的编号而改用自己设计的编号标志，其中 label 参数可以为任何能用作计数器显示的文本；\tag*命令使得编号严格按照 label 的内容排版，不带任何可能会由文档类自动放置的修饰符号（如括号）。\tag 和\tag*命令还可以用于所有星号版的 amsmath 对齐环境。例如：

$$x^2 + y^2 = z^2 \qquad (5.44)$$
$$x^3 + y^3 = z^3$$
$$x^4 + y^4 = r^4 \qquad (*)$$
$$x^5 + y^5 = r^5 \qquad *$$
$$x^6 + y^6 = r^6 \qquad (5.44')$$

```
\begin{gather}
    x^2+y^2 = z^2 \label{eq:r2} \\
    x^3+y^3 = z^3 \notag \\
    x^4+y^4 = r^4 \tag{$*$} \\
    x^5+y^5 = r^5 \tag*{$*$} \\
    x^6+y^6 = r^6 \tag{\ref{eq:r2}$'$}
\end{gather}
```

注意在上面的例子中使用了\label 和\ref 命令对公式进行下一层编号。如果 amsmath 包使用了 leqno 选项，公式编号将打印在公式的左边。缺省时编号在公式的右边。

5.5.2 公式计数器的复位

在标准 LaTeX 中如果我们想使公式在分节内部编号，即第一节中的公式编号为(1.1)、(1.2)、...，第二节中的公式编号为(2.1)、(2.2)、...等等，我们可能需要重新定义\theequation 命令：

```
\renewcommand{\theequation}{\thesection.\arabic{equation}}
```

但是即使这样还得在每一个分节的开头使用命令将公式计数器复位。为了使这类工作更方便一些，amsmath 引入了\numberwithin 命令，使用下面的命令就可以使公式编号自动与分节编号捆绑在一起，且能够随着新的分节的开始自动复位：

```
\numberwithin{equation}{section}
```

正如命令的名字所隐含的意思那样，\numberwithin 也可以用于除公式计数器以外的别的计数器，不过结果不一定都尽如人意。要提醒大家注意的是，能用标准 LaTeX 命令的场合尽量不要用 AMS-TeX 命令。

为了更容易地交叉引用到一个公式，AMS-TeX 提供了\eqref 命令，例如要引用到标号为 e:baset 的公式，可以使用命令\eqref{e:baset}。

5.5.3 次级编号顺序

amsmath 包还提供了一个 subequations 公式环境，这个环境中可以很方便地在特定次级编号机制的组内对公式编号。例如：

```
\begin{subequations}
    ...
\end{subequations}
```

假设此前的公式编号为 4.8，则上述环境中的公式将依次被编为 4.9a, 4.9b. 4.9c, ...。如果在\begin{subequations}命令后面立即使用\label 命令，产生的\ref 将引用到当前的父公式编号(4.9)而不是 4.9a。subequations 环境同时使用 parentequation 和 equation 这两个公式编

号计数器，二者都可以用标准 LaTeX 命令\addtocounter, \setcounter 和\value 等计数器命令改动。另外次级编号的风格由标准 LaTeX 方法控制，如下面重新定义\theequation 命令后，公式编号将变成罗马字母方式：

```
\begin{subequations}
\renewcommand{\theequation}{\theparentequation \roman{equation}}
...
```

5.5.4 数学模式下间隔的微调

尽管 TeX 在排版数学模式下的公式时将元素之间的间隔已经处理得比较好了，有时候我们还是需要对其中的某些地方进行微调。4.6 节中我们介绍过几个标准 LaTeX 中控制空白间隔的命令，下面再介绍几个 AMS-TeX 所增加的命令，这些命令不论是全拼格式还是缩写格式也都可在数学环境外部使用。

表 5.9 AMS-TeX 扩充的间隔控制命令

正空格（拉宽）			负空格（紧缩）		
全拼格式	缩写格式	含义	全拼格式	缩写格式	含义
\thinspace	\,	增加较小间距	\negthinspace	\!	稍微紧缩
\medspace	\:	增加中等间距	\negmedspace		紧缩中等幅度
\thickspace	\;	增加较大间距	\negthickspace		大幅度紧缩

另外还有一个命令\mspace{len}，其参数 len 代表间隔的大小，单位为 LaTeX 数学长度单位。一个数学长度单位（记作 mu）等于 1/18em(相当于\quad 命令所产生的间隔)，因此为了产生朝左跳过一个\quad 大小的间距，可以使用\mspace{-18.0mu}命令。

5.5.5 amsmath 包的选项及子包

amsmath 包和类识别一些可选项设置，这些选项可以影响数学运算符的运算上下界或标志的位置：

(1) centertags: 缺省值，split 环境中的标志放置在环境总高度的垂直居中处。
(2) tbtags: “顶部或底部标志”，split 环境的标志文本将放在最后一行（如果编号放在公式右边）或第一行（如果编号放在公式左边）。
(3) intlimits: 同 sumlimits 相似，只是作用于积分符号。
(4) nointlimits: 缺省值，与 intlimits 设置相反。
(5) namelimits: 缺省值，与 sumlimits 相似，只是作用于特定的“运算符名”，如 det, inf, lim, max, min 等，这些运算符在公式中出现时下面会放置下标内容。
(6) nonamelimits: 与 namelimits 设置相反。

(7) sumlimits: 缺省值，将求和符号上下的内容放在与求和符号垂直的位置。这个选项的设置同时影响其他同类型符号（如$\prod$，$\coprod$，$\otimes$，$\oplus$等），但是不包括积分符号。

(8) nosumlimits: 即使是在公式显示中，也将上下内容放在求和类符号的侧面的上下方位置

下面 3 个选项是影响整个文档的全局性选项，因而在\documentclass 命令中设置，但是只有当使用\usepackage 命令装入 amsmath 包之后才能被识别：

(9) leqno: 将公式编号放置在公式的左边。

(10)reqno: 缺省值，将公式编号放置在公式的右边。

(11)fleqn: 将公式放在离左边注区固定缩进量的位置，而不是放在文本栏的中央。

AMS-LaTeX 包括了一个部件集合，这些部件可以用\usepackage 命令各自独立地装入，最值得关注的恐怕要算 amsmath 包了，但是其他包也可以单独使用。注意当 amsmath 包被装入时，系统同时自动装入 amsbsy, amsopn 和 amstext 三个包：

amsmath: 定义附加的多行显示公式环境及许多对数学模式进行加强的命令。

amsbsy: 定义\boldsymbol 和\pmb 命令。

amsopn: 提供\DeclareMathOperator 命令，用来定义如\sin 和\lim 一样的新的运算名。

amstext: 提供\text 命令，用于在公式环境中插入一段文本内容。

上面这些包我们在前面都曾经提到过，这里集中起来重新说明了一遍。其他的包都必须显式装入，而且大部分我们都还没有介绍，下面进行完整描述：

amscd: 通过引入 CD 环境定义了一些命令，使用这些命令可以更方便地产生换向图。该包不支持斜向箭头。参见 5.3.4 节。

amsintx: 为产生积分符号与求和符号提供更富有描述性的命令格式。

amsthm: 为\newtheorem 命令提供证明环境和功能扩充。详见 5.6 节。

amsxtra: 提供一些诸如\fracwithdelims 和\accentedsymbol 命令。参见 5.2.4 节。

upref: 使\ref 命令打印的交叉引用编号永远用正体罗马字体而不管上下文如何。

最后我们介绍几个随 AMSFonts 发布的包：

amsfonts: 定义了\mathfrak 和\mathbb 命令，建立了数学环境中使用的 msam（附加数学符号 A）、msbm（附加数学符号 B，黑板粗体）、eufm(Euler Fraktur)、特殊大小的 cmmib（粗数学斜体和粗小写希腊字母）以及 cmbsy（粗体数学符号和草书）等字型字体。

amssymb: 定义 AMS 字体集中的所有数学符号的名称。此包装入 amsfonts 包。

eufrak: 建立 Fraktur 字母集合。

eucal: 使\mathcal 命令使用 Euler 草体来代替通常使用的 Computer Modern 草体。

5.5.6 AMS-LaTeX 文档类

随 AMS-LaTeX 包还发布了 amsart 和 amsbook 这一对文档类，分别与标准 LaTeX 的 article 和 book 相对应。这两个文档类原本是用于准备向 AMS 投稿写手稿用的，当然没人禁止你使用它们去干别的事。使用这两个文档类文件时系统将自动装入 amsmath 包，因而只需在文档开头放置\documentclass{amsart}或\documentclass{amsbook}命令，就可以使用 amsmath 包的数学扩充了。

5.6 对 theorem（定理）环境的扩充

AMS-LaTeX 所提供的 amsthm 包扩充了标准 LaTeX 的\newtheorem 命令，同时对标准 LaTeX 的 theorem 机制也进行了扩展，允许通过指定风格来操纵定理的外观布局。

这里“定理”的概念和环境可以用于任何带标号的一段阐述，一般用额外的空白与主文本分开且使用不同的字体，如我们通常所说的数学定理、推论、假设、定义以及评注等等都属于这里的“定理”实例。这些结构以标注（如 Theorem 或 Remark）加上一个编号开头，编号在所有同标注的结构中顺序编排。

使用中我们经常需要调整定理环境的布局（如为了符合不同数学期刊的投稿版面格式），另外不同类别的“定理”可能需要不同的排版格式（如评注和定义用 roman 字体而数学定理用 italic 字体）。

5.6.1 定义新的定理环境

本节中介绍的定义定理环境的命令只能放在文档的导言区或包文件中。

(1) 定义新的定理环境

同标准 LaTeX 版一样，\newtheorem 命令定义一个新的定理式结构。两个必选参数分别代表新环境的名称及环境中文档内容的标注名；一个可选参数指定环境的编号方法，缺省使用自己单独的计数器。如：

\newtheorem{*env-name*}{*label-text*}

定义名字叫做env-name的定理环境，输出的文档内容用label-text名来标识，使用自己单独的计数器。

\newtheorem{*env2-name*}[*env-name*]{*label-text2*}

定义名字叫做env2-name的定理环境，输出的文档内容用label-text2名来标识，与env-name环境使用同一个计数器。

\newtheorem{*env3-name*}{*label-text3*}[*section*]

定义名字叫做env3-name的定理环境，输出的文档内容用label-text3名来标识，使用的计数器在section代表的分节内部计数，即每次遇到新的\section命令计数就重新从1开始，编号的显示也由分节计数器的值加上定理计数器的值组合而成。

(2) 定理环境的风格选择

\theoremstyle{*style*}

所有定理类结构的外观风格都可以用这条命令定义，用\newtheorem 定义的任何定理环境都将按照定义时使用\theoremstyle 命令设置的当前风格排版。下面的控制序列控制 Cor 环境按照 break 风格排版而 Exa 环境按照 plain 风格排版，此后定义的新定理环境也使用 plain 风格，直到出现另一条\theoremstyle 命令：

```
\theoremstyle{break} \newtheorem{Cor}{Corollary}
\theoremstyle{plain} \newtheorem{Exa}{Example}[section]
```

如果想限制\theoremstyle命令的作用域，可以用一对花括号将作用范围包括起来。表5.10列出了可选的定理环境风格。

表 5.10 定理环境风格一览表

风 格 名	含 义
Plain	模拟标准 LaTeX 的定理环境风格，另外使用了两个参数 \theorempreskipamount和\theorempostskipamount
Break	定理标题后面带一个换行
marginbreak	具有break风格特点，另外将定理编号放在边注区
changebreak	具有break风格特点，但是将定理标题中的文字和编号位置互换
Change	定理标题中的文字和编号位置互换，但后面不带换行
Margin	将定理编号放在边注区，但标题后面不带换行

(3) 定理环境的字体选择

\theorembodyfont{*font-declarations*}

定理体的字体选择与\theoremstyle 定义的风格完全独立，实践证明这样处理有很多好处 我们可以使用\theorembodyfont 命令来设置定理体字体，如

{\theorembodyfont{\rmfamily} \newtheorem{Rem}{Remark}}

定义了一个定理环境 Rem，使用当前定理风格，定理体选择 rmfamily 字体。同定理环境的风格选择相似，定理体字体的选择取决于环境用\newtheorem 命令定义时最后出现的 \theorembodyfont 命令。如果没有使用\theorembodyfont 命令或使用了带空参数的 \theorembody{}命令，新的定理环境将采用缺省的定理体字体（除了 plain 外的所有定理风格都将\normalfont\slshape 作为缺省字体）。

\theoremheaderfont{*font-declarations*}

这条命令用来设置定理标题中的字体，因为它是全局性的设置，所以要求文档导言区中最多只能出现一条\theoremheaderfont 命令。如果确实需要不同的定理标题字体，我们只好定义新的定理风格并提供所需要的字体参数。

(4) 定理环境的前后间距

两个附加的参数将影响定理环境周围的垂直间距：\theorempreskipamount 命令和 \theorempostskipamount 命令分别定义了环境之前和之后的空白。这两个参数可以用于所有的定理环境并且能够用标准的长度宏来控制。它们的值可以用\setlength 命令设置。它们是橡皮长度，因此可以包含加减号。

5.6.2 定理环境的定义和使用实例

假设文档导言区中包含下面的声明：

```
\theoremstyle{break} \newtheorem{Cor}{Corollary}
\theoremstyle{plain} \newtheorem{Exa}{Example}[section]
{\theorembodyfont{\rmfamily} \newtheorem{Rem}{Remark}}
\theoremstyle{marginbreak} \newtheorem{Lem}[Cor]{Lemma}
```

```
\theoremstyle{change}
\theorembodyfont{\itshape} \newtheorem{Def}{Cor}{Definition}
\theoremheaderfont{\scshape}
```

则控制序列

```
\begin{Cor}
    This is a sentence typeset in the theorem
    environment \Lenv{Cor}.
\end{Cor}
\begin{Exa}
    This is a sentence typeset in the theorem
    environment \Lenv{Exa}.
\end{Exa}
\begin{Rem}
    This is a sentence typeset in the theorem
    environment \Lenv{Rem}.
\end{Rem}
\begin{Lem}[Ben User]
    This is a sentence typeset in the theorem
    environment \Lenv{Lem}.
\end{Lem}
\begin{Def}[Very impressive Definition]
    This is a sentence typeset in the theorem
    environment \Lenv{Def}.
\end{Def}
```

将产生如下的输出结果：

COROLLARY 1
This is a sentence typeset in the theorem environment **Cor**.

EXAMPLE 5.6.1 *This is a sentence typeset in the theorem environment* **Exa**.

REMARK 1 This is a sentence typeset in the theorem environment **Rem**.

2 LEMMA (BEN USER)
This is a sentence typeset in the theorem environment **Lem**.

3 DEFINITION (VERY IMPRESSIVE DEFINITION)
This is a sentence typeset in the theorem en-

vironment **Def.**

最后的两个例子中使用了定理环境的可选参数，这个参数将被括号括起来排在定理标题的后面

5.7 数学风格参数

本节中介绍如何控制全局性的数学公式排版风格以及怎样改变公式或子公式中特定元素的尺寸大小等等。

5.7.1 字符大小的控制

如果字母和数学符号出现在分式或上下标中时，它们就会变小。实际上 TeX 有 8 种不同的公式处理风格，它们的名称、命令和含义分别如下：

D, D'	\displaystyle	数学模式下一行中的正常公式；
T, T'	\textstyle	文本中的内置公式；
S, S'	\scriptstyle	作为上下标的公式；
SS, SS'	\scriptscriptstyle	上上标（次级上标）或下下标（次级下标）公式。

文本模式中的公式（处于一对$或\(...\)中间）使用T风格；而\[...\]之间的公式将使用D风格。TeX还使用了三种不同大小的数学符号尺寸，分别是文本尺寸、上下标尺寸和次级上下标尺寸。公式中不同部分的字符大小可以用下面的原则来确定：

D, D', T, T'风格的元素使用文本尺寸；

S, S'风格的元素使用上下标尺寸；

SS, SS'风格的元素使用次级上下标尺寸。

下面这个连续的分式例子很好地展示了各种字符风格：

$$b^0 + \cfrac{a^1}{b_1 + \cfrac{a^2}{b_2 + \cfrac{a^3}{b_3}}}$$

```
\normalsize
\[ b^0 + \frac{a^1}{b_1 +
\frac{a^2}{b_2 +
\frac{a^3}{b_3}}}
\]
```

我们可以改变缺省的风格，使数学公式更加美观一些（注意为了节省输入时间我们给 \displaystyle 命令定义了一个缩写形式\D ）：

$$b^0 + \cfrac{a^1}{b_1 + \cfrac{a^2}{b_2 + \cfrac{a^3}{b_3}}}$$

```
\newcommand{\D}{\displaystyle}
\normalsize
\[ b^0 + \frac{a^1}{\D b_1 +
\frac{a^2}{\D b_2 +
\frac{a^3}{b_3}}}
\]
```

5.7.2 LaTeX 数学风格参数

我们前面提到过，LaTeX 提供了两个可选参数 leqno 和 fleqn，leqno 使公式的编号放

在公式左边，fleqn 使公式在离左边注固定距离(\mathindent 大小)的地方左对齐。另外 LaTeX 提供了许多代表长度的数学风格参数，除了\jot 和\arraycolsep 外都是橡皮长度，这些参数的名称和含义如下：

(1) \arraycolsep: array环境中两列之间水平间隔宽度的一半，缺省为5pt。

(2) \jot: eqnarray或eqnarray*环境中两行之间额外增加的垂直间距，缺省为3pt。

(3) \mathindent: fleqn选项定义的公式缩进宽度（从左边注算起），缺省时等于第一级列表项的缩进量(2.5em)。

(4) \abovedisplayskip: 指定不带fleqn选项的长公式显示时顶部预留的垂直空白高度（带fleqn选项时使用\topsep），所谓长公式系指其左端离左边注的距离小于上一行末端离左边注的距离的公式；缺省为12pt+3pt–9pt = 6pt。

(5) \belowdisplayskip: 指定不带fleqn选项的长公式显示时下面预留的垂直空白高度（带fleqn选项时使用\topsep），缺省为12pt+3pt–9pt = 6pt。

(6) \abovedisplayshortskip: 指定不带fleqn选项的短公式显示时顶部预留的垂直空白高度（带fleqn选项时使用\topsep），所谓短公式系指从上一行末端位置的右边开始显示的公式；缺省为0pt+3pt = 3pt。

(7) \belowdisplayshortskip: 指定不带fleqn选项的短公式显示时下面预留的垂直空白高度（带fleqn选项时使用\topsep），缺省为7pt+3pt – 4pt = 6pt。

第六章　图形与表格

6.1　图形(picture)环境

LaTeX提供了专门的图形环境以及一些绘图命令，利用这个环境可以进行简单的作图工作（如画直线、圆形和曲线等等）。为了使用图形环境及其中的绘图命令，首先要弄清楚LaTeX绘图时的坐标单位和坐标系。

6.1.1　坐标系

一般我们取图形的左下角作为坐标原点，水平朝右的方向作为x轴，垂直朝上的方向作为y轴，LaTeX提供了长度命令\unitlength来代表坐标的单位长度（缺省值为1pt），而图形中的每个点都对应惟一的一个坐标(x,y)，其中x代表此点离y轴的距离为x*\unitlength，y代表此点离x轴的距离为y*\unitlength。同一般的长度命令一样，我们可以使用\setlength命令在图形环境开始之前设置坐标的单位长度\unitlength，一旦\unitlength的值发生变化，图形环境中的图形将按比例缩放。

由于坐标原点为图形的左下角，所以坐标值一般都是正值，但是LaTeX也并不禁止使用负坐标值。同数学中的概念相似，x坐标为负表示该点在y轴的左边；y坐标为负表示该点在x轴的下面。

6.1.2　图形环境

LaTeX提供了picture环境用于绘制简单的图形：

```
\begin{picture}(x, y)
    ......
\end{picture}
```

其中(x, y)代表图形环境右上角的坐标，也就是说它决定了图形的大小或作图的范围。一般我们在进入图形环境之前设置坐标的单位长度（通常先设置成整数长度，待调试好以后再改变此设置值，直到得到满意的图形尺寸为止），如：

```
\setlength{\unitlength}{1.0cm}
\begin{picture}(15,10)
......
\end{picture}
```

图形环境中可以使用（也只能使用）LaTeX提供的绘图命令（见6.1.3节）、字体字号命

令以及设置线宽的命令。注意不能在图形环境内部使用\setlength命令来改变坐标的单位长度

6.1.3 绘图命令

1.定位命令

\put(*x*, *y*){*elememt*}

\multiput(*x*, *y*)(*dx*, *dy*){*number*}{*element*}

这两条命令将一个图形元素分别在指定的位置绘制多遍。参数element代表用后面介绍的绘图命令所绘制的图形元素；(x, y)代表element的放置位置（指的是基准点位置，注意不同的图形元素代表其位置的基准点可能不同）；(dx, dy)表示下一次绘制图形元素时放置位置与上一次的相对位移；number表示总的绘制次数。换句话说，后面这条定位命令可以用下面的控制序列来模仿：

```
\put(x, y){elememt}
\put(x+dx, y+dy){elememt}
.....
\put(x+(number-1)*dx, y+(number-1)*dy){elememt}
```

2.文本输出命令

图形环境中也可以输出用作标注的文本，LaTeX支持几种方法来实现这一点：直接定位输出文本，使用文本盒命令，使用minipage环境，使用文本模式下的\fbox命令

(1) 直接输出

此时只要将文本的内容作为\put或\multiput命令的element参数，就可以在指定位置放置文本了。如：


```
\begin{picture}(33,15)
\put(13,7){\vector(1,1){5}}
\put(18,12){An arrow}
\end{picture}
```

(2) 文本盒命令

LaTeX在图形环境下可以使用的文本盒命令有\parbox，\framebox，\makebox和\dashbox等几条，下面分别予以论述。

\parbox[*pos*]{*size*}{*text*}

产生段落模式文本盒，size 为文本盒宽度，text 为要显示的文本。可选参数 pos 指定文本的基准点：为 b 时取文本盒中最后一行文本的左下角位置；为 t 时取文本盒中第一行文本的左下角；不用（缺省）时取文本盒左边界的垂直中点。

\framebox(*width,height*)[*pos*]{*text*}

这条命令将文本 text 放入一个方形区域，区域的边界用一个实线边框包围。区域的宽度和高度分别为 width 和 height（单位均为\unitlength）。文本在矩形中的放置位置由可选参数 pos 指定（缺省时输入文本水平、垂直都居中）：

t　水平居中，垂直方向与盒子的顶边对齐；
b　水平居中，垂直方向与盒子的底边对齐；
l　垂直居中，水平方向与盒子的左边对齐；
r　垂直居中，水平方向与盒子的右边对齐；
s　垂直居中，水平方向伸展以充满整个盒子。
tl　位于盒子左上角；
tr　位于盒子右上角；
bl　位于盒子左下角；
br　位于盒子右下角。

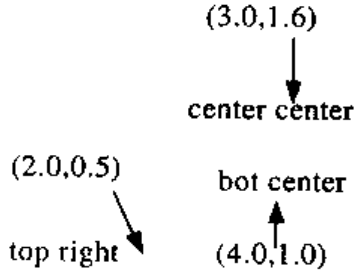
注意后四种取值里字母的顺序是无关紧要的，如[tl]等同于[lt]。

\dashbox{*dwid*}(*width, height*)[*pos*]{*text*}

与\framebox 功能相似，只是边框不是实线而是虚线，dwid 参数表示边框中的短线段长度（单位也是\unitlength）。

\makebox(*width,height*)[*pos*]{*text*}

与\framebox 功能相似，只是没有边框。\makebox 的宽度和高度经常设置为 0，这样就可以把文本放在图形的任何地方，且如果必要的话可以得到文本重叠的效果。如：




```
\setlength{\unitlength}{1cm}
\begin{picture}(5,3)
\put(3,1.6){\makebox(0,0){center center}}
\put(2,0.5){\makebox(0,0)[tr]{top right}}
\put(4,1.0){\makebox(0,0)[b]{bot center}}
\put(2,2.8){\makebox(0,0)[l]{flush left}}
\put(4.0,0.5){\vector(0,1){0.5}}
\put(4.0,0.45){\makebox(0,0)[t]{(4.0,1.0)}}
\put(1.5,1){\vector(1,-1){0.5}}
\put(1.5,1.05){\makebox(0,0)[b]{(2.0,0.5)}}
\put(3.0,2.1){\vector(0,-1){0.5}}
\put(3.0,2.15){\makebox(0,0)[b]{(3.0,1.6)}}
\put(1.5,2.8){\vector(1,0){0.5}}
\put(1.45,2.8){\makebox(0,0)[r]{(2.0,2.8)}}
\end{picture}
```

\framebox, \dashbox 和\makebox 所产生的文本盒的基准位置都是盒子的左下角。

(3) minipage环境

\begin{minipage}[*pos*]{*size*}
　text
\end{minipage}

其作用相当于\parbox[pos]{size}{text}命令。

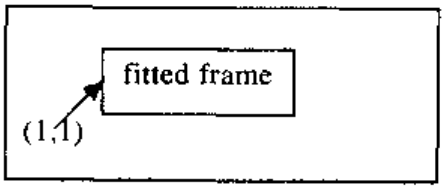
\parbox命令或minipage环境所输出的文本，既可以独立用于图形环境中，也可以放在前面介绍的文本盒当中，不过如果属于后一种情况，必须注意文本盒内部的\parbox命令或

minipage环境的pos参数与文本盒命令的pos参数要兼容：如果文本盒的pos使用了b或t，则内部的\parbox命令和minipage环境也必须使用；如果文本盒的pos使用r或t或缺省值，则内部的\parbox命令和minipage环境都必须使用缺省值（即不给出pos参数）。

(4) 文本模式下的\fbox命令

\fbox{*text*}

这条命令在文本模式下使用时在文本text的四周添加一个方框，它也可以在图形环境中使用 \framebox命令绘制的文本边框尺寸预先知道，但其中的文本与边框的距离不好控制；\fbox命令的边框正好贴着文本的外轮廓，文本与边框之间的距离可以通过长度命令\fboxsep加以控制。\framebox和\fbox命令往往可以相互配合使用，如：




```
\setlength{\unitlength}{1cm}
\begin{picture}(5,2)
\setlength{\fboxsep}{0.25cm}
\put(0,0){\framebox(5,2){}}
\put(1,1){\fbox{fitted frame}}
\put(0.5,0.5){\vector(1,1){0.5}}
\multiput(1,1)(0.18,0){3}{\line(1,0){0.09}}
\put(0.5,0.5){\makebox(0,0)[t]{\small (1,1)}}
\end{picture}
```

3.层叠文本输出

\shortstack[*pos*]{*x**yy**zzz*}

将参数中\\分隔的各个部分用一列多行的方式堆叠。这条命令在文本模式下也可以使用。pos表示对齐方式，可以取l（左对齐），r（右对齐）或c（缺省值，居中对齐）注意堆叠的层数并不受限制。基准点在整个堆叠区域（即文本所占用的最小矩形区域）的左下角 例如如下的输出结果

This			L	
spacing	Not	Single	e	
leaves	really	characters	t	b
some	the	place much	t	e
room	best	better	e	s
for	for words	than	r	t
help	at all	words	s	

对应的源文档控制序列为

```
\setlength{\unitlength}{1cm}
\begin{picture}(9.6,3)
\put(1.0,0.5){\shortstack[l]{This\\spacing\\leaves\\some\\ room \\ for \\ help}}
\put(3.0,0.5){\shortstack{Not\\really\\the\\best\\ for words \\ at all}}
\put(5.0,0.5){\shortstack[r]{Single\\characters\\ place much \\ better \\ than \\ words}}
\put(8.0,0.8){\shortstack{L\\e\\t\\t\\e\\r\\s}}
```

```
\put(9.0,1.2){\shortstack{b\\e\\s\\t}}
\end{picture}
```

4.画线及线宽命令

```
\thinlines
\thicklines
```

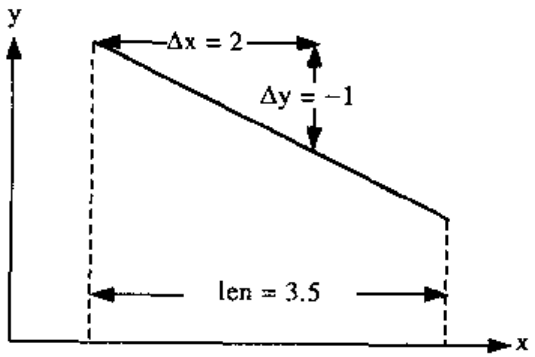
这两条命令分别选择细线和粗线，进入图形环境时缺省选择\thinlines 。它们影响\line, \vector, \circle和\oval命令。

```
\linethickness{len}
```

这条命令设置此后的画线宽度为len，如\linethickness{1mm}。该命令影响\line, \framebox以及\dashbox命令。

```
\line(x, y){len}
```

自\put或\multiput命令所定位置开始画一条直线，直线的斜率由(x, y)确定，长度由len确定。如果x=0则表示直线与y轴平行，此时直线长度为len（单位为\unitlength）；否则直线斜率为y/x，直线在x轴上的投影长度为len（单位为\unitlength）。例如：




```
\setlength{\unitlength}{0.1cm}
\begin{picture}(45,37)
\thinlines
\put(1,10){\vector(1,0){40}}
\put(1,10){\vector(0,1){21}}
\put(9,10){\dashbox{1.5}(0,23){}}
\put(39,10){\dashbox{1.5}(0,8){}}
\thicklines
\put(9,33){\line(2,-1){30}}
\thinlines
\put(16.5,14){\vector(-1,0){7.5}}
\put(31.5,14){\vector(1,0){7.5}}
\put(12,33){\vector(-1,0){3}}
\put(22,33){\vector(1,0){3}}
\put(25,30.5){\vector(0,1){2.5}}
\put(25,27.5){\vector(0,-1){2.5}}
\put(17,33){\makebox(0,0){\scriptsize x=2}}
\put(25,29){\makebox(0,0){ &
        \scriptsize  y=-1}}
\put(24,14.5){\makebox(0,0){ &
        \scriptsize {len}{}=3.5}}
\put(41,10){\makebox(0,0)[l]{x}}
\put(1,32){\makebox(0,0)[b]{y}}
\end{picture}
```

5.箭头命令

```
\vector(x, y){len}
```

在\put或\multiput命令所定位置开始画一条直线，并在终点处绘制一个箭头，(x, y)和len的含义同\line命令完全一样。例如：

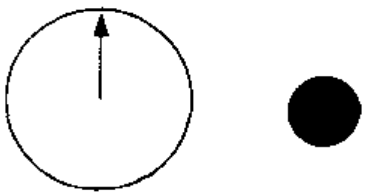
```
\setlength{\unitlength}{1cm}
\begin{picture}(5,2)
\thicklines
\put(5,0){\vector(-1,0){5}}
\put(0,0){\vector(1,1){2}}
\put(2,2){\vector(3,-2){3}}
\end{picture}
```

6.作圆命令

\circle{*d*}

\circle*{*d*}

这两条命令分别在\put或\multiput命令所指定的位置绘制一个圆，前一条命令绘制空心圆，后一条（星号版）命令绘制实心圆。参数d表示圆的直径。所绘制的圆的基准点处于圆心，例如：



```
\setlength{\unitlength}{1cm}
\begin{picture}(3,1.6)
\put(1,1){\circle*{0.2}}
\put(1,1){\circle{1.2}}
\put(1,1){\vector(0,1){0.6}}
\put(2.5,1){\vector*{0.5}}
\end{picture}
```

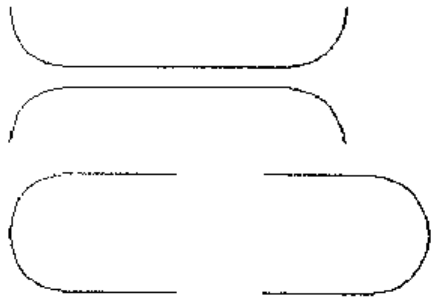
7.画圆角矩形命令

\oval(*x*, *y*)[*opt*]

绘制一个长为x个坐标单位、宽为y个坐标单位的圆角矩形，其基准点为矩形几何中心点。每个角都是1/4圆周，圆周的半径尽量取大并能保证各个矩形边的光滑连接。可选参数opt的可能取值及其含义如下：

l	绘制圆角矩形的左半边；
r	绘制圆角矩形的右半边；
t	绘制圆角矩形的上半边；
b	绘制圆角矩形的下半边。
lt, tl	绘制圆角矩形的左上1/4；
rt, tr	绘制圆角矩形的右上1/4；
lb, bl	绘制圆角矩形的左下1/4；
rb, br	绘制圆角矩形的右下1/4。

注意如果使用了opt可选参数，圆角矩形的基准点和尺寸都不变，只是其中的一部分可见而另一部分不可见而已。例如：



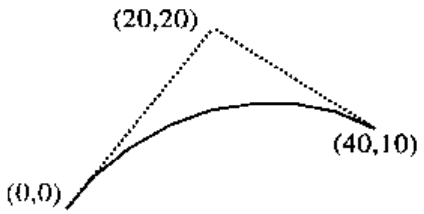
```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,2)
\put(1.75,4.2){\oval(3.5,1.2)[b]}
\put(1.75,2.6){\oval(3.5,1.2)[t]}
\put(1.75,1.0){\oval(3.5,1.2)[lt]}
\put(3.25,1.0){\oval(3.5,1.2)[rb]}
\end{picture}
```

8.画 bézier 曲线命令

\bezier{*number*} $(x_1, y_1)(x_2, y_2)(x_3, y_3)$

\qbezier[*number*] $(x_1, y_1)(x_2, y_2)(x_3, y_3)$

这两条命令都能够在(x_1,y_1)点到(x_3,y_3)点之间绘制一条bézier曲线，(x_2,y_2)为曲线控制点，number参数表示曲线所经过的路段数目，换句话说加上两个端点曲线总共由number+1个点组成。第二条命令中将number视为可选参数，如果不选，则由计算机确定最佳点数。例如：

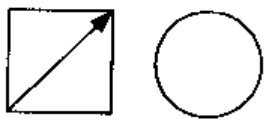



```
\setlength{\unitlength}{0.1cm}
\begin{picture}(55,20)
\qbezier(10,5)(30,25)(50,15)
\put(10,5){\circle*{0.8}}
\put(30,25){\circle*{0.8}}
\put(50,15){\circle*{0.8}}
\put(10,5){\line(1,1){1}}
\multiput(10,5.3)(1,1){20}{.}
\multiput(30,24.7)(1,-0.5){20}{.}
\put(3,5){\makebox(0,0)[lb]{\small (0,0)}}
\put(27,25){\makebox(0,0)[r]{\small(20,20)}}
\put(55,12){\makebox(0,0)[r]{\small(40,10)}}
\end{picture}
```

9.画边框命令

\frame{*element*}

这条命令在\put命令或\multiput命令所指定的位置绘制一个矩形边框，其内部再绘制任何类型的图形元素element，边框的基准点位于左下角。例如：



```
\setlength{\unitlength}{1cm}
\begin{picture}(2.5,1.2)
\put(0,0){\frame{\vector(1,1){1.0}}}
\put(2,0){\frame{\circle{1.0}}}
\end{picture}
```

6.1.4　图形环境的嵌套

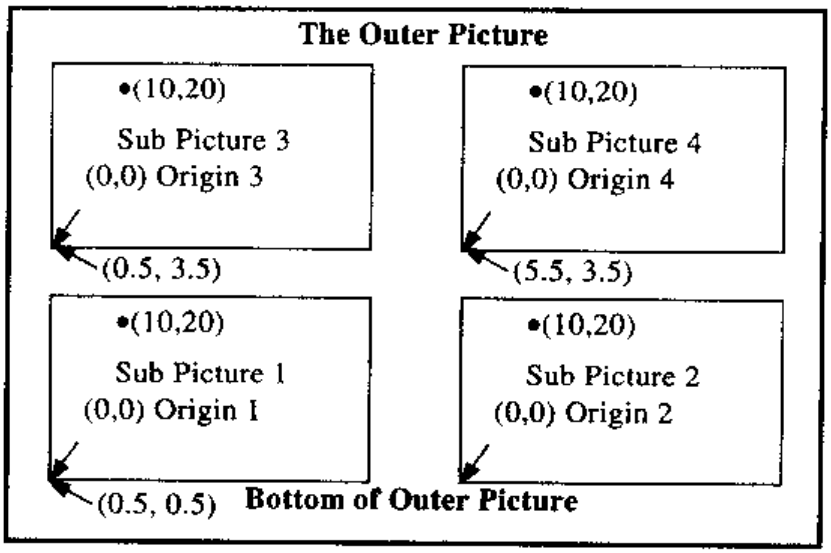
在一个图形环境中可以再包括更小的图形环境，即可以将一个完整的图形环境作为外

层图形环境中一个\put或\multiput命令中的element参数，此时内层图形环境的原点将作为\put命令的基准点。内部环境可以采用不同的单位坐标长度（即使用\setlength{unitlength}{ul}命令），如果不显式设置，将采用外层图形环境的单位坐标长度。例如控制序列

```
{\setlength{\unitlength}{1cm}
\begin{picture}(10.0, 6.6)
\thicklines
\put(0, 0){\framebox(10.0, 6.6){}}
\put(5.0, 6.3){\makebox(0, 0){\bfseries The Outer Picture}}
\thinlines
\put(0.5, 0.5){\setlength{\unitlength}{1mm}
\begin{picture}(50, 25)
\put(0, 0){\framebox(40, 25){Sub Picture 1}}
\put(10, 20){\circle*{1}}
\put(10.2, 20){\makebox(0, 0)[l]{(10, 20)}}
\put(4, 4){\vector(-1, -1){4}}
\put(5, 5){\makebox(0, 0)[lb]{(0, 0) Origin 1}}
\end{picture}}
\put(5.5, 0.5){\setlength{\unitlength}{1mm}
\begin{picture}(50, 25)
\put(0, 0){\framebox(40, 25){Sub Picture 2}}
\put(10, 20){\circle*{1}}
\put(10.2, 20){\makebox(0,0)[l]{(10, 20)}}
\put(4, 4){\vector(-1, -1){4}}
\put(5, 5){\makebox(0, 0)[lb]{(0, 0) Origin 2}}
\end{picture}}
\put(0.5, 3.5){\setlength{\unitlength}{1mm}
\begin{picture}(50, 25)
\put(0, 0){\framebox(40, 25){Sub Picture 3}}
\put(10, 20){\circle*{1}}
\put(10.2, 20){\makebox(0, 0)[l]{(10, 20)}}
\put(4, 4){\vector(-1, - 1){4}}
\put(5, 5){\makebox(0, 0)[lb]{(0, 0) Origin 3}}
\end{picture}}
\put(5.5, 3.5){\setlength{\unitlength}{1mm}
\begin{picture}(50, 25)
\put(0, 0){\framebox(40, 25){Sub Picture 4}}
\put(10, 20){\circle*{1}}
\put(10.2, 20){\makebox(0, 0)[l]{(10, 20)}}
\put(4, 4){\vector(-1, -1){4}}
\put(5, 5){\makebox(0, 0)[lb]{(0, 0) Origin 4}}
\end{picture}}
\put(5, 0.1){\makebox(0, 0)[b]{\bfseries Bottom of Outer Picture}}
\thicklines
\put(1, 0.25){\vector(-2, 1){0.5}}
\put(1.1, 0.25){\makebox(0, 0)[l]{\small (0.5, 0.5)}}
```

```
\put(1, 3.25){\vector(-2, 1){0.5}}
\put(1.1, 3.25){\makebox(0, 0)[l]{\small (0.5, 3.5)}}
\put(6, 3.25){\vector(-2, 1){0.5}}
\put(6.1, 3.25){\makebox(0, 0)[l]{\small (5.5, 3.5)}}
\end{picture}
```

输出结果为：



6.1.5　图形元素的存储

\newsavebox{*sb*}

这条命令定义了一个新的、可以存储图形元素的矩形容器sb。

\savebox{*sb*}(*width, height*)[*pos*]{*element*}

这条命令将图形元素element存于容器sb中，其中(width, height)和pos参数的含义同6.1.3节中\makebox命令的相应参数一样。这条命令可以在图形环境外部甚至文档的导言区中使用，从而产生全文档范围可用的图形元素。

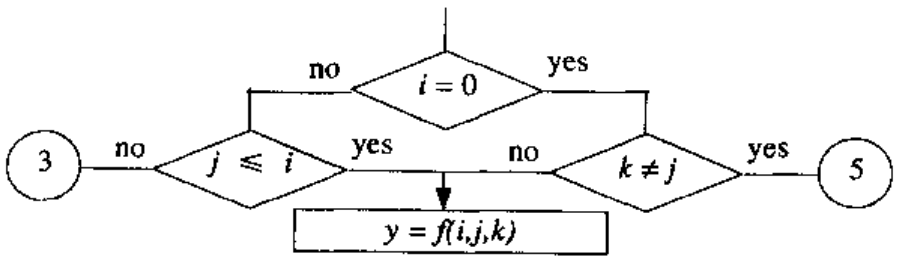
\usebox{*sb*}

这条命令将调用sb中所存储的图形元素。例如控制序列

```
\setlength{\unitlength}{1cm}
\begin{picture}(10,3)
\newsavebox{\testcon}
\savebox{\testcon}(0,0){%
    \thicklines
    \put(0,0.5) {\line(-2, -1){1.0}}
    \put(0,0.5) {\line(2, -1) {1.0}}
    \put(0, -0.5){\line(-2,1) {1.0}}
    \put(0, -0.5){\line(2,1)   {1.0}}
    \put(0,0.5) {\makebox(0,0)[b]{\put(0,0){\line(0,1){0.5}}}}
    \put(1.0,0) {\line(1,0){1.0}}
    \put(-1.0,0){\line(-1,0){1.0}}
```

```
        \put(-1.1,0.1){\makebox(0,0)[br]{no}}
        \put(1.1,0.1){\makebox(0,0)[bl]{yes}}}
\thicklines
\put(5,2){\usebox{\testcon}\makebox(0,0){$i=0$}}
\put(3,1){\usebox{\testcon}\makebox(0,0){$j\le i$}}
\put(7,1){\usebox{\testcon}\makebox(0,0){$k\neq j$}}
\put(0.5,1){\circle{1.0}\makebox(0,0){3}}
\put(9.5,1){\circle{1.0}\makebox(0,0){5}}
\put(5,1){\vector(0, -1){0.5}\circle*{0.1}}
\put(3.5,0){\framebox(3,0.5){$y = f(i,j,k)$}}
\end{picture}
```

输出结果为：



也可以用\savebox命令存储一个完整的图形环境，这时\savebox命令的width和height参数将被忽略，因为图形元素的尺寸在所保存的图形环境中有相似的尺寸参数。这时命令格式为

```
\savebox{sb}{\begin{picture}(width, height)
                ... ... ...
                \end{picture} }
```

6.2 表格(tabular)环境

6.2.1 表格环境定义

```
\begin{tabular}[pos]{cols}
    textlines
\end{tabular}
\begin{tabular*}{width}[pos]{cols}
    textlines
\end{tabular*}
```

tabular 环境可以用来排版可带有横线和竖线的表格，LaTeX 自动确定表列的宽度。tabular*环境同 tabular 环境相似，只是可以用 width 参数指定表格的整体宽度，另外 textlines 表示的行中必须在第 -列后面包含一个合适的@表达式（见下文）。

由于 LaTeX 是用一个垂直盒子来放置一个表格，所以表格可以同其他盒子或文本进行水平对齐定位。如为了使表格在文档页面内水平居中，可以如下将表格环境包括在定位环境之间：

```
\begin{center}
    \begin{tabular}{...}
        ......
    \end{tabular}
\end{center}
```

6.2.2　表格环境参数格式

1.pos 可选参数

该参数表示表格相对于外部文本行基线的位置：取 t 表示表格顶部与当前外部文本行的基线重合；取 b 表示表格底部与外部文本行的基线重合；不使用或缺省时表格按照外部文本行的基线垂直居中。

2.cols 必选参数

该参数指明表格的格式：l 表示列中的文本左对齐，r 表示列中的文本右对齐，c 表示列中的文本居中；p{width}指定列的文本宽度为 width，列中文本将按照该宽度自动换行；|表示画单竖线，||表示画双竖线。

3. textlines 格式

textlines 中的每一行文本（以\\命令结束）代表表格中的一行，文本中的&符号表示输出点跳至表格下一列。

6.2.3　表格文本行中的命令

1.\tabularnewline 命令

这条命令可以用于 tabular 或 array 命令，强制一表格行的结束。而\\命令除了可以结束整个一行表格内容外，还可以在单个列的内容中实现换行。

2.\hline 命令

这条命令只能位于第一行前面或紧接在行结束命令\\的后面，表示插入一根横线。例如：

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

```
\begin{tabular}{rll} \hline
7C0 & hexadecimal \\3700 & octal \\
11111000000 & binary \\ \hline \hline
1984 & decimal \\ \hline
\end{tabular}
```

Welcome to Boxy's paragraph. We sincerely hope you'll all enjoy the show.

```
\begin{tabular}{|p{4.7cm}|} \hline
Welcome to Boxy's paragraph.
We sincerely hope you'll all enjoy the show.\\
\hline \end{tabular}
```

3.\cline{*n-m*}命令

这条命令的放置位置同\hline 命令，并且在一行中可以出现多次。该命令从第 n 列的左边开始，画一条到第 m 列右边结束的水平线。

4.\vline 命令

该命令画一条竖直线，其高度等于所在位置的行高。用这个命令可以得到那些不贯穿整个表格的垂直直线

5.@表达式

@{*text*}

@表达式在出现的两列中间每一行上插入文本 text，同时去掉原来在这两列间自动插入的空白。如果我们需要继续使用空白，必须在@表达式的 text 参数中包含\hspace{}命令。如果希望某两个特定列之间的间隔与确省的标准间隔不同，可以在表格环境的 textlines 参数中相应的位置放上@{\hspace{*new_hspace*}}控制，此时该处列间间隔将变成 new_hspace。

@表达式中可以使用\extracolsep{*colsep*}控制，将后面所有列间间距在原有标准间隔的基础上增加 colsep 大小。在 tabular*环境的 textlines 参数中，必须使用@{\extracolsep\fill}命令，这样才能调整后面的列间距使得整个表格能够达到指定的总宽度。

一个表格即使左右边界没有竖线或其他表征符号，相应位置与后（前）面的列之间也会插入等于标准列间隔一半的空白。如果不希望有这些空白，可以在 textlines 的开始或结尾处使用@{}表达式。

下面的例子说明了@表达式的一个常见用途：解决数字的对齐问题。因为没有简单现成的办法将数字列按小数点位置对齐，我们可以用 2 个列来“骗过”系统：一列放右对齐的整数部分，另一列放左对齐的小数部分，然后在\begin{tabular}中使用@{.}命令以便用“.”代替两列间的符号，从而产生只有一列按照小数点位置对齐的数据的视觉效果。

no leading space	\begin{tabular}{@{}l @{}} \hline no leading space\\ \hline \end{tabular}
leading space left and right	\begin{tabular}{l} \hline leading space left and right\\ \hline \end{tabular}

6.\multicolumn 命令

\multicolumn{*num*}{*col*}{*text*}

这条命令只能位于一行的开始或者一个列分隔符(&)的后面，把接下来的 num 个列合并成一个列处理，其内容为 text。该列的总宽度等于各个合并前的列的宽度之和加上列间间距之和。col 参数的含义同 tabular 环境中的 cols 参数相似。

下面的例子中还用\multicolumn 命令在数字表列的上面放置了一个列标。

Pi expression	Value
π	3.1416
π^{π}	36.46
$(\pi^{\pi})^{\pi}$	80662.7

```
\begin{tabular}{c r @{.} l}
Pi expression &
\multicolumn{2}{c}{Value} \\ \hline
$\pi$ & 3&1416 \\ $\pi^{\pi}$ & 36&46
\\ $(\pi^{\pi})^{\pi}$ & 80662&7 \\
\end{tabular}
```

6.2.4　表格样式参数命令

在表格的生成中，\LaTeX 要利用许多样式参数，来设置其标准值。用户可以使用 \setlength 命令改变这些值，既可以在导言中进行全局性改动，也可以只局限于一个环境中，但不能在表格内部对其进行改动。

1.\tabcolsep 命令

用于 tabular 和 tabular*环境，表示两列间标准间隔的一半大小。

2.\arraycolsep 命令

用于 array 环境，也相当于两列间间隔的一半大小。

3.\arrayrulewidth 命令

代表表格中水平线与垂直线的宽度。

4.\doublerulesep 命令

代表表格中使用垂直竖线时两根竖线间的距离。

5.\arraystretch 命令

代表表格中行间距的缩放比例因子（确省的标准值为 1）。

6.2.5　表格示例

这个例子描述了一个有框表格的空白表。这儿的困难在于设置空白盒子的高度和宽度，因为通常这些量是由文本条目自动确定的。表格的 LaTeX 源文档为：

```
\newsavebox{\kk}\newsavebox{\kkk}
\sbox{\kk}{\framebox[4mm]{\rule[0mm]{3mm}}}
\sbox{\kkk}{\usebox{\kk}\usebox{\kk}\usebox{\kk}}
\begin{tabular}{|l|c|c|c|}\hline
  \multicolumn{4}{|c|}{\rule[-0.3cm]{0mm}{0.8cm}\bfseries
      Budget Plan 2001--2003}\\
  \hline\hline
  \rule[-0.4cm]{0mm}{1cm}Project
  & \multicolumn{3}{|l|}{Nr. \usebox{\kkk}\hspace{0.5cm}
 \vline\hspace{0.5cm}Name\usebox{\kkk}\usebox{\kkk}\usebox{\kkk}
   \usebox{\kkk}}\\ \hline
  \multicolumn{1}{|r|}{Year} & 2001 & 2002 & 2003 \\
  \cline{2-4}
  & (GB \pounds) \vline\ US \$ & (GB \pounds) \vline\ US \$
  & (GB \pounds) \vline\ US \$ \\ \hline
Investment & \hspace{2.5cm} & \hspace{2.5cm} & \hspace{2.5cm} \\
Costs      & & & \\ \hline
Operating & & & \\
Costs      & & & \\ \hline
Industrial& & & \\
Contract & & & \\ \hline
\multicolumn{4}{|l|}{\rule[-1.2cm]{0mm}{1.5cm}Signature
```

```
        \hspace{5cm}\vline~Authorization} \\ \hline
    \end{tabular}
```

表格最终的输出情况如下：

Budget Plan 2001-2003

Project	Nr.		Name			
Year	2001		2002		2003	
	(GB £)	US $	(GB £)	US $	(GB £)	US $
Investment Costs						
Operating Costs						
Industrial Contract						
Signature				Authorization		

6.3 可浮动图形和表格

6.3.1 可浮动体的定义和使用

现在的出版物中大多数都含有许多图形和表格，这些元素不允许出现跨页，因此必须特别对待。一种办法是每当出现一个当前页余下空间排不下的图形或表格时就换一新页，但这会使页面留下部分空白，因而非常难看。更好的解决办法是将这些排不下的图表“浮动”起来，LaTeX 提供 2 种环境定义这种可浮动体，一个用于图形的浮动，另一个用于表格的浮动。为了更好地利用这两个环境，我们有必要大概了解一下 LaTeX 内部是如何处理这种浮动的，否则这个问题将成为你使用 LaTeX 的一个主要障碍，你可能会发现 LaTeX 总是不能把浮动的图表放到你所期望的位置。

下面就是 LaTeX 定义浮动图表的环境：

```
\begin{figure}[placement specifier]
\begin{table}[placement specifier]
```

figure 环境和 table 环境中的任何文档都将作为可浮动体对待。两个环境都支持一个叫做 placement specifier 的字符串可选参数，它的作用是告诉 LaTeX 可浮动体允许移动到的位置，其中的每个字符代表了一种允许浮动的位置，表 6.1 列举了所有的可用字符及其

含义

表 6.1　　允许浮动位置字符及其含义一览表

字符	允许浮动的位置
h	只允许放在环境开始位置，适合于较小的图表
t	允许放在页的顶端
b	允许放在页的底端
p	允许放在一个专门只放置可浮动体的页中
!	忽略禁止可浮动体放置的内部参数（如每页可放浮动体的最大数量）的作用

例如我们可以使用下面的环境开始定义一个表格：

\begin{table}[!hbp]

参数[!hbp] 允许 LaTeX 将该表格放在当前位置(h)，或者放在某一页的底部(b)，或者放到专门放置浮动图表的一页(p)，即使难看也无所谓(!)。如果不给出 placement specifier 参数，其缺省值为[tbp]。

LaTeX 将尽量按照文档作者所指定的位置放置所遇到的可浮动体。如果当前页放不下一个浮动图表，该图表将被放到一个图形队列或表格队列中等候放置，这两种队列都是先入先出(FIFO)队列。当 LaTeX 开始一个新页时，首先考察两种队列中的浮动图表能否填满一个专门的浮动图表页，如果不可能，则从两个队列中取出最先出现（即位置应当最靠前）的浮动体，然后按照它指定的允许放置位置再次放置（h 除外，因为此时 h 已经肯定不起作用了） LaTeX 严格维护着可浮动体出现的先后次序，在某个图形没有被放置妥当以前，其后出现的所有图形也都将跟着往后拖延放置。因此提醒读者注意：如果浮动图表放置的位置不如你意愿，经常是某个别的浮动图表影响了两个浮动图表队列中的一个。

关于浮动图表还有一些命令。用下面的命令可以给浮动图表定义标题：

\caption[*short caption*]{*caption text*}

LaTeX 在指定的标题文字 caption text 之前，自动加上 “Figure”或 “Table”以及一个动态累计的图形或表格编号。

下面两条命令同建立内容目录表的\tableofcontents 命令相似：

\listoffigures　　\listoftables

前者产生文档中所有图形的目录列表，后者产生文档中所有表格的目录列表。这些列表中将使用浮动图表的标题全名，因此如果你给图表起的标题太长的话，在用\caption 命令指定如表标题时，必须用 short caption 可选参数给出一个短的名字，目录列表中将采用短名字来代替长标题。文档中用\label 和\ref 命令可以建立到浮动图表的索引。

下面的例子画一个方框并将它插入到文档之中，如果你想保留一块地方以便以后向排版好的文档中贴图，可以使用类似的方法。

```
Figure~\ref{white} is an example of Pop-Art.
\begin{figure}[!hbp]
    \makebox[\textwidth]{\framebox[5cm]{\rule{0pt}{5cm}}}
```

```
\caption{Five by Five in Centimetres} \label{white}
\end{figure}
```

上面的例子中，LaTeX 将尽全力把一幅空图放在当前位置。如果放不下则放在当前页的底部，如果还放不下则看是否需要另起一页放此图及从表格列表中取出的足够多的待放表格。如果又没有足够多的待放浮动图表形成专门的一页，LaTeX 就重起一页并再次试图放置，就好像该图的环境在新页的开始处发生一样。

有些情况下我们可能有必要使用下面这两条命令：

```
\clearpage        \cleardoublepage
```

前者让 LaTeX 立刻放置两个浮动体队列中的所有浮动图表并重起一页；后者用于双面打印的文档，除了完成前者的功能外，新页必须是左首页（也就是说有时可能还要空上一页）。

有时我们可能不希望在某些页面上（如文档标题页）出现可浮动体，或者不希望可浮动体放置在正文中第一次引用它的位置之前，这样就可以使用下面这条延迟浮动的命令：

```
\suppressfloats[placement specifier]
```

这条命令确保在当前页中由 placement specifier 指定的区域不再放置可浮动体。注意此命令只是抑制从命令放置位置到本页结束之间文档中定义的浮动图表，因此浮动图表队列中以前的可浮动体仍旧可能出现在当前页上。

6.3.2 影响浮动的样式参数

1.可浮动体数量计数器

这类参数可以用\setcounter 命令来设置新的值。

topnumber: 一个页面顶部最多可放置的浮动图表的数量；

bottomnumber: 一个页面底部最多可放置的浮动图表的数量；

totalnumber: 一个页面中最多可放置的浮动图表的总数量（任何位置）；

dbltopnumber: 双栏文档中可放置的横跨两栏的浮动图表的最大数量。

2.浮动区域比例

这些参数都表示一个不大于 1 的小数的命令，可以用\renewcommand 命令改变它们的值。

\topfraction: 页面顶部多大部分（占整个页面的比例）可以放置浮动图表；

\bottomfraction: 页面底部多大部分（占整个页面的比例）可以放置浮动图表；

\textfraction: 页面中多大部分（占整个页面的比例）必须用文本填充，因此\topfraction +\bottomfraction 必须不大于 1-\textfraction ；

\floatpagefraction: 浮动图表页面中可浮动体所占页面区域的最小比例；

\dbltopfraction: 用于双栏文档页面，含义同\topfraction 参数；

\dblfloatpagefraction: 用于双栏文档页面，含义同\floatpagefraction 参数。

3.可浮动体周围的空白

这些参数都是表示橡皮长度的命令，因此可以用\setlength 命令来改变它们的值。

\floatsep: 页面顶部或底部相邻的可浮动体间的垂直间隔；

\textfloatsep: 页面顶部或底部的可浮动体与文本之间的垂直间隔；

\intextsep: 页面中间的可浮动体与其前后文本之间的垂直间隔；

\dblfloatsep: 用于双栏文档，含义同\floatsep 参数；

\dbltextfloatsep: 用于双栏文档，含义同\textfloatsep 参数。

4.可浮动体放置前后执行的命令

这些参数代表页面顶部可浮动体放置之后或页面底部可浮动体放置之前LaTeX自动执行的命令，通常我们可以不使用它们。有时可以用\renewcommand 命令来设置它们，用它们来输出浮动表格与正文之间的隔离标尺，但这里加入的标尺必须是零高度。

\topfigrule: 在页面顶部浮动图表之后被执行；

\botfigrule: 在页面底部浮动图表之前被执行；

\dblfigrule: 用于双栏文档，含义同\topfigrule 参数。

这三条命令缺省时什么都不做，但我们可以如下定义，以便在页面顶部浮动图表与文本之间输出 0.4pt 粗细的标尺：

```
\renewcommand{\topfigrule}{\vspace*{-3pt}
    \rule{\columnwidth}{0.4pt}\vspace{2.6pt}
```

注意上面控制序列中加入的标尺高度是 0 (−3 + 0.4 +2.6) pt，满足\topfigrule 命令对标尺零高度的要求。

上面所有专门用于双栏文档的参数影响的都是横跨两栏的浮动图表。双栏文档中也可以使用不分栏页面中影响浮动的参数，不过此时它们影响的只是填充某一栏的可浮动体。

第七章 LaTeX2e 类和包的设计

LaTeX 是一种文档预处理系统，使得文档的书写人员能够将精力集中到文档的内容（逻辑结构）而不是文档的格式上，例如使用\chapter{title}定义一章的开始，而不用想着该选择 18pt 大小的粗体字等问题。如何将文档的逻辑结构转换到排版格式，所需的信息存放在文档类文件中。另外，有些文档特性（如颜色或包含的图像等）则与文档类相独立，单独放在包文件中存储。

早期的 LaTeX（如 LaTeX2.09）对用户书写类和包文件提供的支持很少，用户不得不使用底层命令。LaTeX2e 提供了许多用于组织包文件的高级命令，同时在已有类或包的基础上建立另一个类或包的工作也变得容易多了，比如我们可以在 article 文档类的基础上建立自己的化学工程系专用技术报告类 cetechr。本章将简单介绍如何书写和组织 LaTeX 的类和包文件。

7.1 类和包的书写

7.1.1 使用 doc 和 docstrip 工具

如果要书写的 LaTeX 类或包比较大，可以考虑使用随 LaTeX 系统发布的 doc 软件。由 doc 书写的类和包有两种使用方法：一是可以在 LaTeX 系统中运行以帮助排版文档；二是由 docstrip 软件加以处理，产生.cls 或.sty 文件。doc 软件可以自动产生定义和命令使用索引以及更新日志列表，这些对大规模 TeX 资源的维护和文档建立来说非常有用。LaTeX 核心的文档资源（如标准文档类等）本身也是 doc 文档，它们以.dtx 文件的形式随 LaTeX 产品发布，实际上用 LaTeX 系统打开 source2e.tex 文件后，我们可以把核心部分的源码当作一个长文档来对待。

7.1.2 文档类和包

在把新的 LaTeX 命令放进文件中之前，首先要确定这些命令是作为一种文档类呢还是作为一种包。处理的一般原则是：如果这些命令可以用于所有的文档类，则把它们处理成一个包；否则把它们处理成一个文档类。

文档类大体可以分为两大类型：一类可以独立存在，类似于 article, report 或 letter；另一类不能独立存在，是别的文档类的扩展或变种，如 proc 文档类是建立在 article 基础上的文档类，因此一个公司可以建立一种内部使用的 ownlet 书信文档类，用来排版输出带有公司标志和地址的信件，这个文档类应当建立在现有的 letter 类的基础上，但是因为不能用于别的文档类，所以我们建立 ownlet.cls 文件而不建立 ownlet.sty 文件。作为对照我们来考察以下图像(graphics)包，这个包中的命令用来将图像插入到 LaTeX 文档中，因为这些命令

可以在各种文档类中使用，所以我们建立的是 graphics.sty 文件而不是 graphics.cls 文件。

7.1.3 命令的名称

LaTeX 的命令可以分为 3 种：一种称为 author 命令，即文档的作者可以使用，如\section, \emph, \times 等等，这些命令的名称大都较短且全由小写字母组成；一种称为类和包书写命令，如\InputIfFileExists, \RequirePackage, \PassOptionsToClass 等等，这些命令的名称大都较长且大小写字母混合使用；还有一种命令称为内部命令，用于实现 LaTeX 系统时使用，如\@tempcnta, \@ifnextchar, \@eha 等等，这些命令的名称中间大都含有@字符，表示它们不能在一般文档中使用，只能在类和包文件中使用。

不幸的是，由于历史原因上述名称上的差别并没有得到严格遵守，比如\hbox 是一条只能在 LaTeX 核心内使用的命令，而\m@ne 却又是一个代表-1 的常数(本应使用\MinusOne 之类的名字)。但是这一命名准则还是有用的：如果命令名称中含有@，则它不是由核心 LaTeX 语言所支持的命令，这样在未来的版本中其行为就可能会发生变化；如果一个命令只是由大小写字母组成，则未来的 LaTeX2e 版本也会继续支持，因此，我们就可以放心使用。

7.1.4 盒子命令与颜色

即使你不准备在文档中使用颜色，但是阅读下面的内容可以帮助你确保你所写的类或包与 color 包相兼容，这样可以让使用彩色打印机的人也能使用你的类或包。

保证与 color 包兼容的最简单的办法，是永远使用 LaTeX 的盒子命令而不要使用 TeX 有关盒子的原语，也就是说用\sbox 命令而不用\setbox 原语，用\mbox 命令而不用\hbox 原语，用\parbox 命令或 minipage 环境而不用\vbox 原语。增加新的可选参数后，LaTeX2e 中的盒子命令的功能已经可以与 TeX 原语相媲美。

下面我们举一个可能会出问题的例子。比如{ttfamily htexti}控制序列中的字体设置是在}之前恢复的，而相似的结构{\color{green} htexti}中的颜色设置却是在}之后恢复的，通常情况下二者之间的差别并无影响，但是考虑下面这个使用 TeX 原语的盒子赋值控制：

```
\setbox0=\hbox{\color{green} htexti}
```

由于颜色恢复是在最后一个}的后面发生的，所以颜色恢复控制并没有存到盒子中，因此所带来的后果取决于颜色是如何实现的，轻则在剩下的文档部分中颜色不对，重则会使支持文档打印的 dvi 驱动程序出错。

7.1.5 定义文本和数学字符

因为 LaTeX2e 支持不同的编码系统，定义产生符号、注音、组合字型等命令时必须使用专门的命令，这些命令在 LaTeX2e 文档的 "Font Selection"中有介绍。LaTeX2e 的这一部分尚在建设中，因此使用时要十分小心。同时定义这类编码独立的命令时应当使用 \DeclareRobustCommand 命令。

注意在数学模式外部再也不能使用数学字体，如\textfont 1 和\scriptfont 2 都不保证在其他模式下有定义。

7.1.6 基本命令

本节介绍 LaTeX2e 提供的一些命令，这些命令可以帮助我们产生结构好、可移植性强的类和包文件

1.装入其他文件

LaTeX 提供下面这些命令用以在一个类或包中使用别的类或包：

\LoadClass　　\LoadClassWithOptions

\RequirePackage　　\RequirePackageWithOptions

出于多个原因，我们强烈推荐使用这些命令装入别的文件而不要使用\input 原语命令：使用\input<filename>命令装入的文件将不被列入\listfiles 列表。如果一个包总是用\RequirePackage, \RequirePackageWithOptions 或\usepackage 命令装入，那么即使装入操作需要多次进行，系统实际上也只装入一次就行了；而形成鲜明对比的是，如果用\input 原语装入一个包，则可能需要系统多次装入，每多装入一次不仅浪费时间和内存，更可能产生奇怪的结果。如果一个包提供了装入时的可选参数，则用\input 装入也可能发生奇怪的结果。

但是如果你向类文件中更新 myclass.sty 文件，则必须保证所有包含\input myclass.sty 命令的旧文件能够继续工作。

2.优化自己的类或包文件

正如前面提到过的那样，书写类或包时尽量多使用LaTeX命令是一个很好的原则，这样可以避免产生意想不到的结果。例如使用\PackageError, \PackageWarning或 \PackageInfo 或等价的类命令，不要使用\@latexerr, \@warning 或\wlog命令。我们仍然可以使用 \ds@<option>定义可选项并用\@options引用它们，但是最好分别使用\DeclareOption和 \ProcessOptions命令，这样你的类或包可以占用较少的内存也更为有效，即更有可能在未来的LaTeX版本中能够继续使用。

3.增强类或包文件的兼容性

尽量使自己的文件能够与环境兼容也很重要，为了做到这一点，文件中应当只包含可见的7位编码文本，文件名也采用8.3格式。另外，文件名当然不能同标准LaTeX发布的产品中的某个文件相同，尽管它们的内容可能很相似。

内部的类或包文件尽量使用一个共同的、富有自然意义的前缀，也是一个好的选择。如武汉大学的文档类或包文件都以whu开头，就可以避免同别的学校的类似文档类或包文件相混淆，例如不这样做的话可能会出现多个学校都有自己的thesis.cls类文件。

如果自己书写的类或包文件依赖于LaTeX内核或标准LaTeX类和包，最好指明所依赖的LaTeX版本的发布日期，例如1994年6月版的LaTeX2e中才引入包出错命令，如果你的包或类使用了该命令，其中必须书写下面的命令：

\NeedsTeXFormat{LaTeX2e}[1994/06/01]

4.几个有用的小工具

以前有些包和文档风格必须重新定义\document或\enddocument命令，现在不用了，我们可以使用\AtBeginDocument和\AtEndDocument命令（详见7.3.6节）。使用这两条命令不仅使你的包或类可以在将来的LaTeX版本中继续使用，还可以使它们更易于同别人的类或包

一起使用。但是要注意\AtBeginDocument中的代码是导言区中的一部分，所以其中放置的内容受到严格限制，特别是任何排版输出命令都是不允许的。

7.2 类或包的结构

类或包文件的大致轮廓由下面这些部分组成：文件标识(identification)部分说明该文件是LaTeX2e包文件或类文件，另外包含对自己的简单介绍；预先声明(preliminary declarations)，部分声明一些命令，这些命令通常只在可选项声明和处理时用到，另外也可以装入其他文件；可选项(options)部分声明和处理可选项；进一步声明(more declarations)部分包括文件的大部分内容，如声明新的变量、命令和字体等，另外装入所有其他必要的文件。

7.2.1 文件标识

类或包文件所做的第一件事情是标识自己。包的标识形式为

\NeedsTeXFormat{LaTeX2e}[*date other-information*]

\ProvidesPackage{*package*}[*date other-information*]

其中package参数说明包的名称。可选参数中date分别表示TeX版本日期和包的完成日期(格式必须是yyyy/mm/dd，而且如果使用可选参数则必须给出date)，other-information用于简单说明包的性质或用途等。例如：

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{latexsym}[1994/06/01  Standard LaTeX package]
```

类的标识形式为

\NeedsTeXFormat{LaTeX2e}

\ProvidesClass{*class-name*}[*date other-information*]

其中class-name参数说明类的名称，可选参数的含义同\ProvidesPackage中的可选参数。例如：

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{article}[1994/06/01  Standard LaTeX class]
```

类的描述信息即other-information在使用类时显示出来，而包的描述信息则存入log文件中。这些信息也可以用\listfiles命令来显示。除了标准LaTeX发布的文件外，所有别的类或包文件的描述信息中都不能像上面两个例子中那样含有Standard LaTeX词组。

7.2.2 其他类和包的使用

LaTeX2e的类和包文件中支持模块化设计，即文件中可以使用更小的模块如别的类或包文件，而不用将所有功能都在一个大的文件中定义。LaTeX类和包文件中可以使用另一个包文件。如

RequirePackage[*options*]{*package*}[*date*]

其中package参数代表要使用的包的名称，option参数是包package能够处理的某些可选项(如果使用了不止一项则用逗号分隔)，date参数表示包package的版本日期。这条命令与author命令\usepackage的格式相同，使得你所定义的包或类中可以使用其他包中所提供的功

能 如

```
\RequirePackage{ifthen}[1994/06/01]
```

可以使你在定义类或包时能够用“if...then...else...”命令，因为这条命令在包ifthen中得到了定义

LaTeX类文件中可以使用另一个类文件。如

```
\LoadClass[options]{class-name}[date]
```

其中class参数代表要使用的类的名称，option参数是类class能够处理的某些可选项（如果使用了不止一项则用逗号分隔），date参数表示类class的版本日期。这条命令与author命令\documentclass的格式相同，使得你所定义的类可以建立在其他类所提供的语法和功能的基础之上。如

```
\LoadClass[twocolumn]{article}
```

此后你所定义的类不用从头开始，只需要做同article类所不同部分的定义，而article类已经定义的东西可以直接使用。

如果你想在使用别的类或包时采用当前类所采用的可选项，通常可以使用下面这两条命令：

```
\LoadClassWithOptions{class-name}[date]
\RequirePackageWithOptions{package}[date]
```

例如：

```
\LoadClassWithOptions{article}
\RequirePackageWithOptions{graphics}[1995/12/01]
```

7.2.3 可选项声明和处理

类和包文件中可以声明可选项并加以处理，如article类提供了可选项twocolumn，我们在前一小节的例子中曾经使用过这一选项。可选项声明的语法格式如下：

```
\DeclareOption{option}{code}
```

其中option参数代表类或包文件所声明和支持的可选项名称，code参数表示当用户使用该可选项打开类或包文件时所执行的控制序列。例如graphics包中的dvips可选项声明是这样实现的：

```
\DeclareOption{dvips}{\input{dvips.def}}
```

这个实现表明每当用户在文档中使用\usepackage[dvips]{graphics}，系统就装入dvips.def文件 再比如article类中定义了a4paper可选项用来设置A4纸张大小的页面，则在它的声明部分要设置两个长度命令\paperheight和\paperwidth：

```
\DeclareOption{a4paper}{%
\setlength{\paperheight}{297mm}%
\setlength{\paperwidth}{210mm}%
}
```

在装入类或包文件时，有时候用户提供的可选项在类或包文件中并没有定义，这时如果使用的是类系统将产生警告，而如果使用的是包将出错。为了避免出错并产生特定的警告信息，书写类或包文件时可以使用下面的声明：

\DeclareOption*{*code*}

这条命令的作用是如果用户使用了没有定义的可选项，则执行code所表示的控制序列，例如fred包中有下面的命令：

```
\DeclareOption*{%
\PackageWarning{fred}{Unknown option `\CurrentOption'}%
}
```

这里\CurrentOption命令代表当前所采用的可选项。这时如果用户在文档中使用了\usepackage[foo]{fred}命令而fred包又没有声明foo可选项，则系统不会出错，但会产生警告信息“Package fred Warning: Unknown option 'foo'”。

所有的可选项声明过以后，为了执行相应的代码，必须使用命令

\ProcessOptions\relax

这条命令将执行所有用户指定的可选项所对应的代码（参见7.3.7节）。例如假设jane包文件包含以下内容：

```
\DeclareOption{foo}{\typeout{Saw foo.}}
\DeclareOption{baz}{\typeout{Saw baz.}}
\DeclareOption*{\typeout{What's \CurrentOption?}}
\ProcessOptions\relax
```

如果用户在文档中书写了\usepackage[foo,bar]{jane}命令，则他会看到下面的信息：

Saw foo. What's bar?

使用下面的命令，可选项可以从一个类或包文件中传递到另一个类或包文件：

\PassOptionsToPackage{*option*}{*package*}

\PassOptionsToClass{*option*}{*class*}

其中option参数代表要传递的可选项名称，package和class参数分别表示接收option的包或类名称。假设我们在article类的基础上书写了一个新类，则用户给出的所有我们没定义的可选项都应当交给article类处理，此时可以使用下面的控制序列：

```
\DeclareOption*{%
\PassOptionsToClass{\CurrentOption}{article}%
}
```

需要注意的是如果采用了上面的方法传递可选项，则目的类或包必须在此后才能装入，否则传递过去的可选项将根本得不到处理，因为包或类文件只有在被装入时才处理可选项。

7.2.4 最小的类文件

书写类或包文件时，最大的工作量在于定义新的命令或改变文档的外观，这些工作使用诸如\newcommand或\setlength之类的命令来完成。另外LaTeX2e还提供了一些书写类和包时专用的命令，我们将在7.3节中加以介绍。

所有的类文件都必须包含三个命令：定义\normalsize，设置\textwidth与\textheight的尺寸。因此最小的类文件可能是如下的形式：

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{minimal}[1995/10/30 Standard LaTeX minimal class]
```

```
\renewcommand{\normalsize}{\fontsize{10pt}{12pt}\selectfont}
\setlength{\textwidth}{6.5in}
\setlength{\textheight}{8in}
```

上面这些正是随LaTeX2e发布的minimal.cls文件的内容。但是这个类文件将不支持脚注、边注、可浮动体等等，同时也不支持两个字母的命令（如\rm），因此绝大多数类文件都要比这个文件复杂。

7.2.5 类文件实例

1.公司内部信件类

一个公司内部可能需要自己特定的信件格式，我们将介绍实现内部信件类所需要的主要框架，而实际实现时可能需要再添加一些细节内容。假设类文件名为neplet.cls。

(1) 进行类文件标识

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{neplet}[1995/04/01 NonExistent Press letter class]
```

(2) 向letter类传递用户使用的可选项，并以A4纸张(a4paper)可选项装入letter类

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{letter}}
\ProcessOptions\relax
\LoadClass[a4paper]{letter}
```

(3) 重新定义firstpage页面类型

为了使用本公司的信件标题和脚注内容，我们重新定义信件首页使用的firstpage页面类型和风格：

```
\renewcommand{\ps@firstpage}{%
\renewcommand{\@oddhead}{newletterhead }%
\renewcommand{\@oddfoot}{newletterfoot}%
}
```

其中newletterhead代表公司信件的标题，newletterfoot代表公司信件的脚注内容。

2.时事通讯类

这个例子在article类的基础上书写一个时事通讯类的框架，文件名叫smplnews.cls。

(1) 进行类文件标识

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{smplnews}[1995/04/01 The Simple News newsletter class]
\newcommand{\headlinecolor}{\normalcolor}
```

(2) 可选项传递与处理

我们将绝大多数可选项都交给article类处理，自己处理onecolumn可选项（将它关掉不用），并定义一个新的可选项green，用以将通讯的标题设置为绿色：

```
\DeclareOption{onecolumn}{\OptionNotUsed}
\DeclareOption{green}{\renewcommand{\headlinecolor}{\color{green}}}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
```

(3) 装入必要的类和包文件

然后我们用twocolumn可选项装入article类：

```
\LoadClass[twocolumn]{article}
```

因为时事通讯使用彩色排版输出，所以我们还要装入color包。为了保证通用性，装入color包时不指定设备驱动可选项，该可选项应当由用户在使用smplnews类时指定。

```
\RequirePackage{color}
```

(4) 重新定义\maketitle命令及分节命令

我们将标题设置为72pt大小的Helvetica bold oblique字体及适当的颜色。

```
\renewcommand{\maketitle}{%
\twocolumn[%
\fontsize{72}{80}\fontfamily{phv}\fontseries{b}%
\fontshape{sl}\selectfont\headlinecolor
\@title
]%
}
```

我们将\section命令重新定义分节格式并关掉分节编号功能：

```
\renewcommand{\section}{%
\@startsection
{section}{1}{0pt}{-1.5ex plus -1ex minus -.2ex}%
{1ex plus .2ex}{\large\sffamily\slshape\headlinecolor}%
}
\setcounter{secnumdepth}{0}
```

(5) 进行三项基本设置工作

```
\renewcommand{\normalsize}{\fontsize{9}{10}\selectfont}
\setlength{\textwidth}{17.5cm}
\setlength{\textheight}{25cm}
```

(6) 其他实际需要的工作

为了真正能够使用，这个类还需要定义文章作者、页面风格等命令。

7.3 类和包书写命令

本节中将详细介绍书写类和包时所使用的命令，其中许多命令在本章前面的内容中有过介绍，这里我们再系统地叙述一遍。

7.3.1 文件标识命令

1.\NeedsTeXFormat{*format-name*} [*release-date*]

这条命令告诉TeX本文件将由format-name格式的TeX系统进行处理，通常我们使用LaTeX2e 可选参数release-date可以用来指定所使用格式的版本日期（必须用yyyy/mm/dd形式），例如也可以指定旧的LaTeX版本，但是系统会出示警告。

2.\ProvidesClass {*class-name*} [*release-info*]

\ProvidesPackage {*package-name*} [*release-info*]

这两条命令分别表示本文件包含文档类class-name和包package-name的定义。其中可选参数release-info表示类或包的说明信息，如果使用则必须采用如下格式：

(1) 要包含yyyy/mm/dd形式的类或包版本日期；

(2) 版本日期后可以跟也可以不跟一个空格加上简短的说明文字，说明文字中可以包括版本号。

上述格式必须严格遵守，这样当使用\LoadClass, \documentclass或\RequirePackage命令装入类或包时，系统才能检查类或包的版本是否太旧。整个release-info的内容将由\listfiles命令显示，所以不宜太长。

3.\ProvidesFile {*file-name*} [*release-info*]

这条命令同ProvidesClass和ProvidesPackage命令相似，只是file-name必须为文件全名（即后缀不能省略）。这条命令可以用来声明除了主类和包文件以外的文件。

7.3.2 文件装入命令

这一组命令可以使我们的文档类建立在别的现有类或包的基础之上。

1.\RequirePackage [*options-list*] {*package-name*} [*release-info*]

\RequirePackageWithOptions {*package-name*} [*release-info*]

在类或包文件中装入别的包。\RequirePackage命令的使用同文档中使用的author命令\usepackage。

2.\LoadClass [*options-list*] {*class-name*} [*release-info*]

\LoadClassWithOptions {*class-name*} [*release-info*]

这两条命令只能在类文件中使用，每个类文件中最多只能使用一次。不能在包文件中使用这两条命令。\LoadClass命令的使用同文档中使用author命令\documentclass装入一个类文件是一样的。

7.3.3 可选项声明命令

1.\DeclareOption {*option-name*} {*code*}

这条命令使得option-name变成所在包或类的一个可选项。code参数是当用户使用了该可选项时要执行的代码，可以包含任何合法的LaTeX2e结构。

2.\DeclareOption* {*code*}

这条命令设置"缺省可选项代码"，指当用户使用的可选项在类或包中没有显式声明时，系统执行该命令的code参数。code参数可以包含任何合法的LaTeX2e结构。

如果类文件中没有设置缺省可选项代码，则将它所接收到的未声明可选项像已声明可选项一样传递给所有包；如果包文件中没有设置缺省可选项代码，则当它接收到未声明可选项后将产生错误。

7.3.4 可选项代码中使用的命令

本节中的命令都只能在\DeclareOption 或\DeclareOption*命令的 code 参数中使用。

1.\CurrentOption

这条命令代表用户指定的当前可选项。

2.\OptionNotUsed

这条命令将用户指定的当前可选项添加到“未使用可选项”列表。

3.\PassOptionsToPackage {*options-list*} {*package-name*}

这条命令将option-list参数中列出的可选项（不止一个时用逗号隔开）传递给包package-name，也就是说此后的\RequirePackage或\usepackage命令可以使用的可选项中包含了option-list中的可选项。例如：

```
\PassOptionsToPackage{foo,bar}{fred}
\RequirePackage[baz]{fred}
```

相当于

```
\RequirePackage[foo,bar,baz]{fred}
```

4.\PassOptionsToClass {*options-list*} {*class-name*}

这条命令只能用于类文件中的\DeclareOption或\DeclareOption*命令的code参数中。它将可选项options-list传递给将要使用\LoadClass命令装入的其他类文件。

我们将\PassOptionsToPackage命令和\PassOptionsToClass命令配合RequirePackage或LoadClass命令的执行效果同7.3.2节中介绍过的两个命令\RequirePackageWithOptions和\LoadClassWithOptions命令的执行结果加以比较：

\RequirePackageWithOptions命令同\RequirePackage命令相似，但是前者在装入指定的包时所使用的可选项列表总是同当前类或包打开时所使用的可选项列表完全相同，而不使用明确给出或由\PassOptionsToPackage命令传递来的可选项。

\LoadClassWithOptions的主要目的是允许一个类简单地继承别的类的特性，如命令\LoadClassWithOptions{article}同控制序列

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
\LoadClass{article}
```

的作用差不多，但是使用\LoadClassWithOptions命令要简单许多，同时执行得也要快一些。如果类声明了自己的可选项，则上面的控制序列与\LoadClassWithOptions命令二者之间就有所不同了，为了进行比较，先看下面两段控制序列：

(1)
```
\DeclareOption{landscape}{\@landscapetrue}
\ProcessOptions\relax
\LoadClassWithOptions{article}
```

(2)
```
\DeclareOption{landscape}{\@landscapetrue}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
\LoadClass{article}
```

如果当前类被装入时使用了landscape可选项，则第一个序列中装入article类时landscape也将得到使用。但是第二个序列中article被装入时根本接收不到landscape可选项，因为只有缺省的可选项传递给了article，而landscape是有显式声明的可选项，不在传递之列。

7.3.5 代码延迟执行命令

1.\AtEndOfClass {*code*}

\AtEndOfPackage {*code*}

这两条命令也主要用于\DeclareOption或\DeclareOption*命令的参数中间，它们将code代码暂时内部保存，等到当前类或包处理完毕后再执行code代码。这两条命令允许多次使用，这时代码参数将按照命令出现的先后顺序存储，最后执行时也按照这个顺序进行。

2.\AtBeginDocument {*code*}

\AtEndDocument {*code*}

这两条命令声明将code代码参数暂时内部保存，分别等到LaTeX执行\begin{document}命令和\end{document}命令时再执行。

\AtBeginDocument命令的code参数将在\begin{document}命令就要执行完毕且字体选择表已经建立起来以后执行，因此可以放置那些需要在排版准备工作一切就绪、当前字体为通常文档字体的上下文环境中执行的代码。code中的代码不应当进行排版工作，因为此时排版的结果很难预料。

\AtEndDocument命令的code参数将在\end{document}命令开始执行、最终页面尚未完成、剩下的可浮动环境还没有处理的时候得到执行。如果code中的某些代码需要在页面和可浮动环境都已经完成排版任务之后才执行，那么这些代码之前必须插入一条\clearpage命令。

这两条命令都允许多次重复使用，此时代码参数将按照命令出现的先后顺序存储，最后执行时也按照这个顺序进行。

3.\AtBeginDvi {*specials*}

这条命令将specials参数保存在一个注册盒子里并写入.dvi文件中文档首页输出开始处。参数中不能带任何排版内容进入.dvi文件。该命令也允许多次重复使用。

7.3.6 可选项处理命令

1.\ProcessOptions

这条命令将执行每个使用的可选项在声明时设定的处理代码。这条命令在类文件和包文件中的执行过程和结果略有不同，我们首先来看在包文件中的情况。

(1) 在包文件中使用

用户或类和包的书写者使用的可选项可以分为局部可选项与全局可选项两种类型：

- 局部可选项

  系指那些在\PassOptionsToPackage{options}命令、\usepackage[options]命令或者\RequirePackage[options]命令的options参数中明确指定的可选项。

- 全局可选项

  系指用户在\documentclass[options]命令的options参数中指定的、除了局部可选项之外的所有可选项。

例如，假设一个文档的开头部分如下：

```
\documentclass[german,twocolumn]{article}
```

```
\usepackage{gerhardt}
```

同时gerhardt包调用了fred包。如

```
\PassOptionsToPackage{german,dvips,a4paper}{fred}
\RequirePackage[errorshow]{fred}
```

则fred包的局部可选项是german, dvips, a4paper和errorshow，而fred包的惟一全局可选项是twocolumn。当执行\ProcessOptions命令时，会发生下面的过程：首先，对于fred.sty目前用\DeclareOption命令声明的每一个可选项，系统考察它是不是一个全局或局部可选项，若是，则按照声明的前后顺序依次执行它们的参数代码；其次，对于未处理的剩余局部可选项，如果它们在别的地方（指除了使用\DeclareOption命令外）被定义了，则系统执行\ds@option命令，否则执行“缺省可选项代码”（如果没有则出错）。所有这些过程都按照可选项声明顺序的先后进行，另外系统保证每个选项的代码最多执行一遍。假设mybook.sty包含控制序列

```
\DeclareOption{dvips}{\typeout{DVIPS}}
\DeclareOption{german}{\typeout{GERMAN}}
\DeclareOption{french}{\typeout{FRENCH}}
\DeclareOption*{\PackageWarning{mybook}{Unknown `\CurrentOption'}}
\ProcessOptions\relax
```

如果在LaTeX源文件中使用命令

```
\usepackage[dvips, german, a4paper, errorshow]{mybook.sty}
```

则用LaTeX编译时会出现如下信息：

```
DVIPS
GERMAN
Package fred Warning: Unknown `a4paper' on input line 9.
Package fred Warning: Unknown `errorshow' on input line 9.
```

其中9表示\ProcessOptions\relax命令在mybook.sty文件中的行号。

(2) 在类文件中使用

在类文件中，\ProcessOptions命令的执行同包文件中一样，只是所有的可选项都当作局部可选项，另外\DeclareOption*的缺省值是\OptionNotUsed而不是出错。

注意\ProcessOptions命令也有星号格式（见下文），不过后跟\relax时最好使用不带星号的版本（正如前面例子中那样），这样可以避免出现一些错误信息。

2.\ProcessOptions* \@options

执行过程同\ProcessOptions命令相似，但是可选项的处理顺序不是按照在类或包中的声明顺序进行，而是由命令调用时的options参数指定。对于包文件来说，全局可选项先处理。

3.\ExecuteOptions {*options-list*}

对于options-list参数中的每一个可选项，这条命令按照顺序依次执行一下\ds@option命令，如果这条命令没有定义，则该可选项将被忽略。option代表要处理的可选项。使用这条命令可以在紧靠\ProcessOptions命令前面的地方提供一个“缺省可选项列表”，例如假设在类文件中我们想把缺省设计设置为双面打印、11pt字体和双栏，则可以使用

\ExecuteOptions{11pt,twoside,twocolumn}

7.3.7 文件操作命令

这类命令处理文件装入问题，可以保证即使用户试图装入不存在的文件，系统也能以友好的界面加以处理。

1.\IfFileExists {*file-name*} {*true*} {*false*}

这条命令可以检测一个文件是否存在，其中 file-name 参数表示待查文件的名字。如果 file-name 文件存在，则执行 true 参数给出的代码；如果 file-name 文件不存在，则执行 false 参数给出的代码。这条命令并不自动装入待查文件。

2.\InputIfFileExists {*file-name*} {*true*} {*false*}

如果 file-name 文件存在，这条命令将执行 true 参数给出的代码并马上装入该文件；如果 file-name 文件不存在，则执行 false 参数给出的代码。这条命令是用\IfFileExists 命令实现的。

7.3.8 报告错误命令

这类命令应当单独在第三方类或包中使用，以负责进行出错报告或者向用户提供相关错误信息。

1.\ClassError {*class-name*} {*error-text*} {*help-text*}

\PackageError {*package-name*} {*error-text*} {*help-text*}

这两条命令用来产生错误信息，出错时将显示error-text参数给出的错误信息以及?提示符，如果用户敲击h键，系统将显示help-text参数指定的帮助文本。在error-text和help-text参数中，可以使用\protect命令来终止当前出错命令的进一步展开；可以使用\MessageBreak命令使显示内容换行；使用\space命令输出空格例如

```
\newcommand{\foo}{FOO}
\PackageError{ethel}{%
Your hovercraft is full of eels,\MessageBreak
and \protect\foo\space is \foo
}{%
Oh dear! Something's gone wrong.\MessageBreak
\space \space Try typing \space <return>
\space to proceed, ignoring \protect\foo.
}
```

将产生如下的输出结果：

```
! Package ethel Error: Your hovercraft is full of eels,
(ethel) and \foo is FOO.
See the ethel package documentation for explanation.
```

如果用户敲击h键，则将再显示下面的内容：

```
Oh dear! Something's gone wrong.
Try typing <return> to proceed, ignoring \foo.
```

2.\ClassWarning {*class-name*} {*warning-text*}

\PackageWarning {*package-name*} {*warning-text*}

\ClassWarningNoLine {*class-name*} {*warning-text*}

\PackageWarningNoLine {*package-name*} {*warning-text*}

\ClassInfo {*class-name*} {*info-text*}

\PackageInfo {*package-name*} {*info-text*}

前4条是警告命令，同上面介绍的两条出错命令相似，只是它们只在屏幕上产生warning-text参数所给出的警告信息，而不显示错误提示，也没有帮助信息。其中第一、二两条命令将显示警告发生处的行号，而第三、四两条命令不显示行号。

后两条命令同前两条命令相似，只是它们将info-text参数所给出的信息只写进log文件（包括行号）。它们没有相应的不产生行号的版本。

在warning-text和info-text参数中，可以使用\protect命令来终止当前出错命令的进一步展开；可以使用\MessageBreak命令使显示内容换行；使用\space命令输出空格。

7.3.9 定义牢固命令

除了前面章节中我们介绍的定义新命令的命令外，LaTeX2e还另外提供了用于类和包文件中的定义命令。其中的星号版命令用来定义对于TeX来说不太长的命令，这样有助于对参数不会成段成段的命令进行错误跟踪。

1.\DeclareRobustCommand {*cmd*} [*num*] [*default*] {*definition*}

\DeclareRobustCommand* {*cmd*} [*num*] [*default*] {*definition*}

这两个命令的参数格式同\newcommand命令一样，但是所定义的命令比使用\newcommand定义的命令要牢固(robust)一些，即使definition参数中的某些代码比较脆弱。所谓牢固命令，是指该命令可以作为其他命令的参数使用，而前面无须使用\protect前缀命令。

使用这两个命令既可以定义新命令，也可以重新定义已有的命令（使现有命令变得更坚挺），如果重新定义已有的命令，系统将在脚本文件中加以记录。假设使用下面的控制序列将\seq命令进行重新定义：

```
\DeclareRobustCommand{\seq}[2][n]{%
\ifmmode
    #1_{1}\ldots#1_{#2}%
\else
    \PackageWarning{fred}{You can't use \protect\seq\space in text}%
\fi
}
```

则尽管\ifmmode命令不能在某些命令参数中使用，但是\seq命令却可以出现在同样的地方，如：

```
\section{Stuff about sequences $\seq{x}$}
```

2.\CheckCommand {*cmd*} [*num*] [*default*] {*definition*}

\CheckCommand* {*cmd*} [*num*] [*default*] {*definition*}

这两条命令的参数格式也同\newcommand命令一样，但是它们不对命令cmd进行定义，而只检查cmd命令的当前定义是否为definition参数所给出的那样，如果检查的结果不一样，则将导致出错。

我们在包中要改变某个命令的定义之前，通常可以用这两条命令检查一下系统的状态，以确定没有其他的包已经将你要改变的命令重新定义过了。

7.4 其他杂类命令

7.4.1 布局参量

\paperheight

\paperwidth

这两个命令表示所用纸张的尺寸参量，通常由文档类设置其值。它们代表的是实际尺寸，而不是像\textwidth和\textheight命令那样只是边注内部主文档区的大小。

7.4.2 大小写字母转换命令

\uppercase {*text*}

\lowercase {*text*}

这两条命令实际上是TeX原语，实现字母的大小写转换。有时候文档类中也使用它们，比如要将标题全部使用大写字母排版等等。但是不幸的是，它们不能转换由某些命令（如\ae或\aa）产生的特殊字符。为此LaTeX提供了下面两条命令来解决这个问题：

\MakeUppercase {*text*}

\MakeLowercase {*text*}

例如：

```
\uppercase{aBcD\ae\AA\ss\OE}
\lowercase{aBcD\ae\AA\ss\OE}
\MakeUppercase{aBcD\ae\AA\ss\OE}
\MakeLowercase{aBcD\ae\AA\ss\OE}
```

由于TeX原语\uppercase和\lowercase将其参数中的所有内容都进行大小写转换（除了用控制序列产生的字符），而其参数中可能包括数学公式、环境名称以及标号等，因此使用后可能会产生意想不到的结果。例如

```
\MakeUppercase{$x+y$ in \ref{foo}}
```

的结果是 $X+Y$ IN LaTeX Warning: Reference 'FOO' on page ... undefined on ...

第八章　图像插入和颜色处理

8.1 引　　言

PostScript 驱动程序的出现使得 LaTeX 能够扩展许多有用的功能，如外部图像文件的插入、图像剪裁和旋转，以及引入彩色等。但是每个驱动程序都有自己的类包文件和用户接口命令，因而所得到的 LaTeX 文档将不得不同特定的驱动程序捆绑在一起。为此 LaTeX2e 通过提供 graphics 和 color 这两类宏包，将所有驱动程序使用的一组命令加以规范化。将用户命令转换成特定驱动程序的代码单独存放在一个.def 文件中，这个文件可以在装入 graphics 和 color 宏包时用选项指定，因此要使用不同的 PostScript 驱动程序时，用户不需要改动文档内容，而只需要改变一下该装入选项即可。

graphics 和 color 包定义的命令可以作为构造模块，可以用来书写更为通用的语法定义。只要新创建的包文件完全基于 graphics 和 color 包，它们就能够支持 graphics 和 color 包所能支持的所有驱动程序。

8.2 外部图像插入和加工

8.2.1 LaTeX 有关图像矩形区域的术语

图像可以放在 LaTeX 的矩形盒子中当作一个文档单元处理，每个矩形盒子在左边界上有一个参考点，通过参考点的水平线称作盒子的基线（如图 8.1 所示）。从图 8.1 中我们可以看出，每个盒子的尺寸由三个长度表示：高度、深度和宽度。高度和深度之和表示了盒子在垂直方向所占据的区域大小，我们称之为总高度。

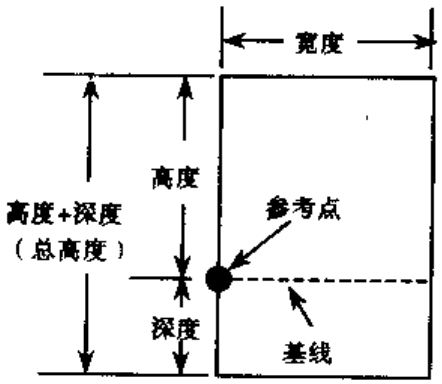


图 8.1　矩形盒子的尺寸和基准术语

对于图像我们可以将包含它的最小矩形区域也可以看作一个盒子，只是参考点以及由

参考点确定的基线按照下面的规则确定：没有经过旋转处理的图像其参考点为图像区域的左下角，因此它的深度为 0；旋转时原来的基准线不变、参考点定义为基线同图像矩形区域左边界的交点（因而可能发生变化）。图 8.2 中的(a)为未旋转过的图像注意其深度为 0；(b)为基线和参考点都未变化但旋转了一定角度的图像；(c)为旋转后基线不变但参考点发生变化的图像，注意其高度为 0。

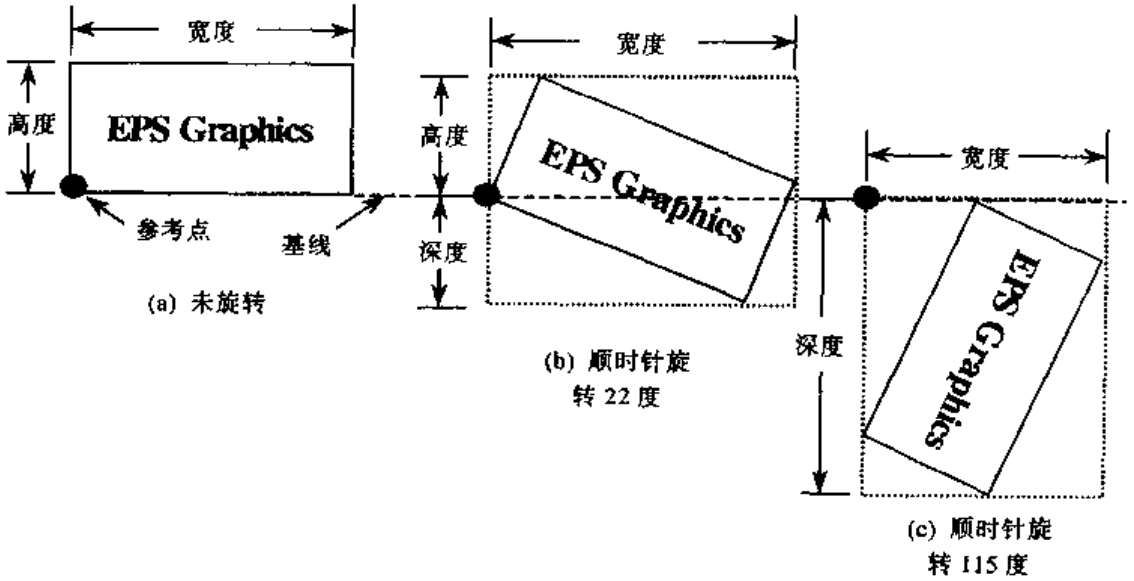


图 8.2 不同旋转角度的图像及其矩形区域

8.2.2 支持外部图像插入的包文件

通常的 LaTeX 系统中都有两个包可以用来操纵外部图像文件，一个是较为基础的 graphics 包，另一个是扩充的 graphicx 包。它们的功能完全相同，只是语法格式不同。这两个包在装入时除了把驱动程序的名字作为可选项外，还可以使用下面这些可选项：

(1) draft: 在插入图像要出现的地方并不真正插入图像，而只是放置一个方框，方框内输出图像文件名。

(2) final: 用来消除 draft 选项的作用，一般用于 draft 选项已经被\documentclass 对应的类文件指定为全局性选项的情况。

(3) hidescale: 在剪裁后的图像所占区域输出空白。

(4) hiderotate: 在旋转后的图像所占区域输出空白。hidescale 和 hiderotate 选项适用于预览器不能处理图像剪裁或旋转的场合。

(5) hiresbb: 查找%%HiResBoundingBox 的值并替换通常用%%BoundingBox 行指定的图像矩形区域的尺寸大小。

8.2.3 使用 graphics 包插入图像

graphics 包提供的基本插入命令为

\includegraphics [*llx, lly*] [*urx, ury*] {*file_name*}

其中 file_name 表示外部图像的文件名，可选参数[llx, lly]代表开始剪裁的左下角坐标，可

选参数[urx, ury]代表结束剪裁的右上角坐标，换句话说它们定义了图像的剪裁范围。坐标单位可以在坐标参数值后直接给出，缺省时使用大像素（big point, PostScript 坐标单位，1/72 英寸）作为坐标单位。

如果只给出一组坐标，则该坐标代表剪裁区域的右上角，左下角当作[0,0]。如果两组坐标都没给出，代表图像不作剪裁全部插入，驱动程序将根据外部图像的格式，通过别的途径获取图像区域的尺寸。例如如果插入常用的紧缩 PostScript 文件（encapsulated PostScript，通常用.eps 后缀），图像区域的大小可以从图像文件本身获取。

\includegraphics 存在一个星号版命令，命令格式及功能相似，只是将不输出任何超出指定区域的图像部分。

8.2.4　使用文本盒命令实现图像缩放和旋转

使用 8.2.1 节中介绍的命令及参数插入的图像都是按照原来的比例和方向正常显示，为了对图像进行缩放和旋转，可以使用有关矩形盒子缩放和旋转的命令，只需在这些命令的 text 参数中使用\includegraphics 命令即可。

1.图像缩放

(1) \scalebox 命令

\scalebox{*h_scale*}[*v_scale*]{*text*}

这条命令将矩形盒子水平和垂直缩放。其中参数 h_scale 代表水平方向的放大倍数；可选参数 v_scale 代表垂直方向的放大倍数，缺省时与 h_scale 相同。

(2) \resizebox 命令

\resizebox{*h_length*}{*v_length*}{*text*}

这条命令可以将插入图像调整到指定的水平和垂直尺寸。其中 h_length 代表水平宽度，v_length 代表垂直高度，二者都可以使用!作为参数值，此时代表该方向按照 1:1 比例确定图像的原始尺寸。

1.图像旋转

(1) 水平倒影

\reflectbox{*text*}

这条命令可以得到原来图像的水平倒影，相当于以图像中心为基点旋转 180 度。

(2) \rotatebox 命令

\rotatebox{*angle*}{*text*}

这条命令可以将图像以盒子基线的左端为基点，沿逆时针方向旋转 angle 度。

8.2.5　使用 graphicx 包插入、缩放和旋转图像

如果用户使用的不是 graphics 包而是 graphicx 包，则可以使用不同的图像插入、旋转以及许多其他功能的带关键字选项的\includegraphics 命令：

\includegraphics [*key=value,…*]{*file_name*}

该命令将 file_name 文件中存放的外部图像插入 LaTeX 文档当前位置，可选的关键字参数 key 指明图像插入时对图像的加工。图像加工关键字分为数值型和标志型（布尔型）两类。数值型关键字（带等号）使用时要在等号后给出具体值；标志型关键字在

\includegraphics 命令中出现即表示其值为 true。可用的关键字有下面这些：

(1) scale=: 图像在原来尺寸上放大的倍数。

(2) width=: 将图像宽度缩放到该参数指定的值（单位按照 LaTeX 方式处理）。

(3) height=: 将图像高度缩放到该参数指定的值（单位按照 LaTeX 方式处理），只能用于未旋转的图像。width=和 height=两个关键字如果只设置了一个，则另一个将同比例缩放（即保持图像的宽高比不变）。

(4) totalheight=: 将图像的高度和深度之和缩放到该关键字指定的值，当图像发生旋转后必须使用该关键字代替 height=。

(5) keepaspectratio: 如果 height=和 width=都进行了设置，则此标志保证维持原图的宽高比不变，而且两个方向上都不超出指定的尺寸设置。换句话说经常只有一个方向上能达到指定的尺寸，而另一个方向上会留下空白。

(6) angle=: 指定图像将逆时针旋转的角度度数（一周 360 度）。此前给出的高度和宽度关键字值也将发生“旋转”，即高度变成了旋转后图像区域的宽度，而宽度变成了高度（正角度旋转）或者深度（负角度旋转）。

(7) origin=: 指定旋转的基点（圆心）。缺省值为 bl，表示左下角，另外可以使用 c（区域中心）、t（顶部）、r（右边）、B（基线）以及它们的组合（如 tr 代表右上角）。

(8) draft: 该标志同 8.2.2 节中的 draft 包文件选项功能相似，但是仅仅对一个图像文件起作用。不显示整个图像，而只显示正确的图像矩形区域边框并在内部显示图像文件名。

(9) clip: 设置该标志将不输出任何超出指定区域的图像部分。

(10) bb=: 值用以空格隔开的 4 个长度，表示显示指定图像的区域。每个长度的缺省单位是大像素点。

(11) viewport=: 同 bb=一样，它的值也代表矩形区域的左下角和右上角，但是坐标值是相对于文档中已有的图像区域的左下角位置，通常用来调整图像显示区域或仅显示图像的一部分。

(12) trim=: 用来减小已有的图像区域。值为以空格隔开的 4 个长度，依次表示将图像区域在左下角 x 方向、左下角 y 方向、右下角 x 方向、右下角 y 方向减小的尺寸。

(13) hiresbb: 该标志的作用同 8.2.2 节中介绍的 hiresbb 选项的功能类似，只是作用于一个图像文件的装入。设置后系统将从图像文件的%%HiResBoundingBox 文本行读入图像尺寸数据。

所有上述关键字都是可选的，而且关键字的数量任意，相邻关键字之间使用逗号隔开。除了 angle=关键字会改变它前面出现的 width=和 height=关键字的含义外，其他关键字的前后顺序无关紧要。

为了同 graphics 包兼容，graphicx 包中也定义了一条对插入图像进行裁剪的 \includegraphics*命令，它的作用等同于使用了 clip 关键字的不带星号版命令。

8.3 PostScript 文件的插入

LaTeX中使用PostScript文件（以下简称ps文件）很简单，只要ps文件中使用适当的注

解行给出图像的矩形区域尺寸即可。也就是说LaTeX要求ps文件必须是诸如xv或xfig软件所产生的eps文件。如果ps文件不满足要求，可以使用ps2epsi程序将文件转换过去。不论怎样得到的ps文件，我们都可以使用如下的控制序列将该文件包含到LaTeX文档中：

```
\documentclass[dvips]{article}
\usepackage{graphicx}
    ......
\begin{figure}[htbp]
    \includegraphics{yourfile.ps}
\end{figure}
```

LaTeX还能处理压缩格式的ps文件，如：

```
\begin{figure}[htbp]
    \includegraphics{yourfile.ps.gz}
\end{figure}
```

8.4　颜色处理

8.4.1　颜色表示

一种颜色可以用预定义的名称表示，也可以用下面这种格式表示：

[model] {specs}

其中 model 参数代表颜色的编码模式，可以是 rgb（红、黄、蓝）组合，cmyk（青、洋红、黄、黑）组合、gray（灰度）或 name（命名）4 种方式中的一种；specs 参数是一个介于 0 和 1 之间的数值，表示模式中相应颜色的强度（亮度）。如[rgb]{1,0,0}表示红色，[cmyk]{0,0,1,0}表示黄色，等等。灰度模型只需要一个数字；命名模型使用 dvips 驱动程序预先定义或建立好的内部颜色名称，当然别的驱动程序不一定使用这些名称。不过 LaTeX 已经将有些颜色命名（红 red，绿 green，蓝 blue，黄 yellow，灰 cyan，蓝绿 megenta，黑 black 和白 white），所有的驱动程序都能识别这些名称。

用户也可以使用下面的命令给某种颜色命名，从而在文档的此后位置用名字来代表这种编码的颜色：

\definecolor*{name}{model}{specs}*

其中 name 参数代表所起的颜色名称，model 和 specs 参数确定颜色的具体编码。

8.4.2　有关颜色的命令

下面这些有关颜色的命令中，所有 col_spec 参数都代表一种颜色，可以是命名颜色如 {blue}，也可以为编码颜色如[rgb]{0,1,0}。

\pagecolor *col_spec*

设置当前及后续页面的背景颜色。

\color *col_spec*

将此后输出的文本的颜色切换到 col_spec 颜色。

\textcolor *col_spec*{*text*}

将 text 参数所代表的文本以 col_spec 颜色输出。

\colorbox *col_spec*{*text*}

将 text 参数所代表的文本输出到一个盒子中，盒子的背景颜色为 col_spec。

\fcolorbox *col_spec1 col_spec2* {*text*}

同\colorbox 命令的功能相似，只是盒子的背景颜色为 col_spec2，而且带一个颜色为 col_spec1 的边框。两种颜色参数要么都是预定义的命名颜色，要么使用同一种编码模式，若为后者编码模式只需要书写一次。如\fcolorbox{red}{green}{Text}或\fcolorbox [rgb]{0,1,0} {0,0,1} {Text}。

\normalcolor

将前景颜色切换到正常颜色。所谓正常颜色，是指文档导言区结束时被激活的前景颜色。因此在导言区中放置一条\color 命令可以改变缺省的标准颜色设置。这种情况同 \normalfont 命令相似。

通常情况下用户最好把文档中要用到的颜色都加以命名，这样可以简化色彩微调时的工作量。当我们打印出文档后，一般都需要将其中某些地方的颜色加以适当调整，因为屏幕输出的颜色同打印机输出的颜色不尽相同，而且同样的颜色设置在不同的打印机上也会得到不同的结果。如果我们对使用的颜色都加以命名的话，这种颜色的微调只需要在颜色命名处进行就可以了，不用找遍整个文档文件。

8.4.3 color 包文件

LaTeX 提供了 color 宏包文件处理文档的颜色，color 包在装入时可以识别如下这些可选项。

- monochrome: 用于预览器不能处理颜色的场合，将所有色彩命令都转换成黑白版。
- dvipsnames: 使 dvips 命名颜色模式（详见 8.4.4 节）适用于别的驱动程序。
- nodvipsnames: 为了节约内存，弃用 dvips 命名颜色模型。
- usenames: 装入所有命名颜色（详见 8.4.4 节）。

8.4.4 颜色命名模式

一种非常有用的颜色编码模式叫做颜色命名模式，它基于 dvips 这种 PostScript 驱动程序所预先定义的 68 种内部命名色彩，如 BurntOrange 和 DarkOrchid 等。使用 dvipsnames 选项装入 color 包后，别的驱动程序将也能够识别这个模式，此时如果用户使用了命令

```
\definecolor{titlecol}{named}{DarkOrchid}
```

则 titlecol 可以作为一种颜色名称出现在前面介绍的各种颜色命令的 col_spec 位置。

如果文档在装入 color 包时使用了 usenames 可选项，则所有 68 种命名颜色都可以用它们本来的名称重新定义，如

```
\definecolor{BurntOrange}{named}{BurntOrange}
```

这样用户在文档中使用这些颜色时将更为方便直观。

为了方便用户使用合适的颜色，可以运行下面这段简单的 LaTeX 文档，并通过打印机输出一张调色板：

```
\documentclass[12pt,a4paper]{article}
\usepackage[dvips]{color}
\usepackage{multicol}
\pagestyle{empty}
\setlength{\oddsidemargin}{0pt}
\setlength{\textwidth}{16cm}
\setlength{\textheight}{22cm}
\setlength{\parindent}{0pt}
\setlength{\parskip}{0pt}
\begin{document}
    \renewcommand*{\DefineNamedColor}[4]{%
        \textcolor[named]{#2}{\rule{7mm}{7mm}}\quad
        \texttt{#2}\strut\\}
    \begin{center}\Large Named colours in \texttt{dvipsnam.def}
    \end{center}
    \begin{multicols}{3}
        \input{dvipsnam.def}
    \end{multicols}
\end{document}
```

要记住，每种打印机输出的颜色都会略有不同，因此最好在每台你可能会用到的打印机上都输出一遍。

第九章 常用的 LaTeX 程序简介

9.1 MikTeX 1.20e

9.1.1 MikTeX 主要特征和组成

MikTeX 是由 Christian Schenk 拥有版权、运行在 Microsoft Windows9x 或 windows NT 上的一种免费 TeX 文档编译软件，1999 年下半年发布的 MiKTeX1.20e 版本主要有以下几个特点：

- 专门为 Win32 设计，支持长文件名和 UNC 文件名；
- 提供安装软件(setupwiz.exe)，使系统易于安装和卸载；
- 文档风格文件、字体等与 TeX3 最新标准(TeXLive3)兼容；
- 对网络工作环境支持较好、如可以集成到多个不同的 TeX 环境、支持 UNC 文件名、支持多个 TEXMF 目录树和本地 TEXMF 目录树以及使用文件名数据库提高文件访问效率等；
- 与 TDS 标准（TeX 目录标准）兼容；
- 提供丰富的字体文件；
- 提供强大的预览功能：支持插入 BMP、EPS、WMF 等多种格式的图像，提供可变放大倍数的放大镜工具，支持 PostScript 特定 DVI，支持虚拟字体(virtual fonts)，可以处理由 Omega 工具生成的 DVI 文件，提供从 dvi 到 PostScript 的打印接口，等等；
- 支持多种发布包装
  - bare.exe: 只包含可执行文件，适合于已经存在 TEXMF 目录树的机器
  - basic.exe: 包括基本应用程序和宏包
  - advanced.exe: 除了同 basic.exe 一样的产品外，还包括 pdfTeX 0.14d、计算机现代 PostScript 字体(Modern PostScript Fonts)和 AMS PostScript 字体(AMS-Fonts PostScript Fonts)
  - complete.exe: 除了同 advanced.exe 一样的产品外，还包括 Omega 1.8、MetaPost 0.641、Texinfo 宏和 Makeinfo 以及 Web 系统(Tangle, Weave 和 Tie)；
- 可自由发布（提供全部源码）。

MikTeX 包括下面这些应用部件：

TeX3.14159: 纯 TeX 编译器；

e-TeX 2.1: 一种 TeX 扩充；

LaTeX: LaTeX 编译器（支持 1999 年 6 月的 LaTeX 规范）；

Yap 0.97: DVI 预览程序；
pdfTeX0.14d: 从 TeX 文档生成 PDF 文件的程序；
dvipdfm 0.12.6e: 将 DVI 文件转换成 PDF 格式文档的工具；
Omega 1.8: 支持 16 位字符集（如 Uni-code）的增强 TeX 版编译器；
METAFONT 2.718: 将字体转换成光栅字体的程序；
MetaPost 0.641: 将图型规格说明转换成 PostScript 命令的程序；
dvips 5.86: 将 DVI 文件转换成 PostScript 文件的工具；
MakeIndex 2.12: 产生文档索引的工具；
BibTeX 0.99c: 产生参考文献的工具；
ps2pk: 将 Type 1 轮廓字体转换并插入 PK 文件的应用；
gif2png: 将 GIF 文件转换成 PNG 文件的工具；
AMS-LaTeX, Babel, PSNFSS, ...多种标准 LaTeX 包；
TcXware, MFware, PSutils, ...许多其他工具。

MiKTeX 的安装过程及使用参见 1.3 节。除了 1.1.2 节中我们介绍的国内站点外，下面再提供几个 MiKTeX 国外下载网站地址：

ftp://ftp.dante.de/tex-archive/systems/win32/miktex/
ftp://ftp.tex.ac.uk/tex-archive/systems/win32/miktex/
ftp://ctan.tug.org/tex-archive/systems/win32/miktex/

9.1.2 MikTeX 配置

1.两类配置文件

MikTeX 使用两个配置文件分开存储系统配置参数：

- 全局配置文件(global configuration file)存放工作站级配置设置，配置文件名为 miktex.ini，位于 TEXMF 根目录的 miktex\config 相对目录下。这些设置将影响所有站上的 MikTeX 用户。
- 个人配置文件(personal configuration file)存放每个用户的系统配置设置，名称用户自己定义，其位置可以存放在用户本地目录中并用本节后文所介绍的配置工具 initexmf.exe 来设置。如果个人配置文件与全局配置文件中有相同参数的设置，系统将使用个人配置文件中的设置。

2.配置文件主要内容

MikTeX 配置文件分成若干命名小节，每一节包含一个特定应用工具或特征所对应的配置参数及其值。下面是一些重要的小节及其主要配置参数。

[MikTeX]

Bin Directories: 指定许多工具（如 makepk.exe 等）的搜索路径，用来定位别的 MikTeX 工具或可执行文件。标准值为 %R\miktex\bin，其中%R 代表 TEXMF 根目录。

Trace: 向控制台输出时的跟踪选择参数，如设置多个值可用逗号分隔。可用的值有 notrace（禁止跟踪），fndb（跟踪文件名数据库），filesearch（跟踪文件查找过程）和 access（跟踪文件访问过程）。

[LaTeX]

Input Dirs: LaTeX 输入文档的搜索路径，标准值为 %R\tex\latex//; %R\tex\generic//。

[Dvips]

DVIPSHEADERS: dvips 头文件（包括.pfb 文件）的搜索路径，标准值为 .;%R\dvips//; %R\fonts\type1//

OVFFONTS: Dvips 工具搜索 OVF 文件的路径，标准值为 .;%R\fonts\ovf//。

TEXCONFIG: Dvips 工具配置文件（如 config.ps）的搜索路径，标准值为 .;%R\dvips//。

TEXINPUTS: 图像文件的搜索路径，标准值为.;%R\dvips//。

VFONTS: 虚拟字体文件（.vf 文件）的搜索路径，标准值为 .;%R\fonts\vf//。

3.initexmf.exe 配置工具

initexmf 是 MikTeX 的配置工具，可以用来定义 TEXMF 根目录列表、更新文件名数据库、更新记忆文件（如 plain.fmt）以及定位个人配置文件等。该工具的使用格式是命令行

initexmf *configure option*

其中必选参数 configure option 表示配置选项，决定配置的项目和设置的值。下面列举一些重要的配置选项，

--dump: 刷新所有记忆文件(*.fmt, *.base, *.mem);

--dump=*program*: 将所有记忆文件同特定的编译器 program 关联，program 必须为 lambda, latex, metafont, metapost, omega, pdflatex, pdftex, tex 等中的一个；

--find-latex-input *FILE*: 在 LaTeX 搜索路径中查找 FILE 文件；

--local-root *root*: 指定 root 为本地 TEXMF 根目录；

--mkpsres: 更新 PostScript 资源数据库'psres.dpr'，可以同后面的—search 选项配合使用；

--mkpsres='dir': 向 PostScript 资源数据库'psres.dpr'中添加一个新的字体目录；

--personal, -p: 不使用个人配置文件；

--personal=*FILENAME*, -p*FILENAME*: 定义个人配置文件名 FILENAME 及其位置；

--report: 产生一个配置报表；

--root-directories *dirlist*, -r *dirlist*: 指定 dirlist 为 TEXMF 根目录列表；

--search: 搜索 PS 资源文件（要求与--mkpsres 配合使用）；

--update-fndb, -u: 刷新整个文件名数据库；

--update-fndb=*root*, -u*root*: 刷新指定 TEXMF 根目录 root 所对应的文件名数据库；

--version, -V: 打印系统版本号并退出。

4.Dvips 工具的配置

同 MikTeX 一同发布的 Dvips 工具，其初始配置如下：

- 产生字体时使用 METAFONT 的 ljfour (HP Laserjet 4)模式；
- 水平分辨率为 600dpi；
- 纸张尺寸为 A4；
- 不使用 CM&AMS PostScript 字体。

为了改变这些设置，可以使用任何文本编辑器打开相对路径\dvips\config\local\下面的 config.ps 文件，其中以 M 开头的行指定 METAFONT 模式，以 D 开头的行指定打印机分

辨率

9.1.3　编译器与工具的使用

本书只简单介绍 TeX 和 LaTeX 两个编译器以及 dvips 和 Yap 两个工具的用法。

1.TeX 的使用

tex　[*options*]　[*firstinputlines*]

options 用来设置 TeX 一些内部参数；firstinputlines 代表第一行输入文本，一般用输入文件名来表示。如

tex　hello.tex

将使 TeX 编译器根据输入文件 hello.tex 产生 dvi 输出文件 hello.dvi。如果输入文件的后缀为.tex、可以省略不写，即上面的命令等同于

tex　hello

但是如果文件全名中包含不止一个圆点，则后缀不再能够省略，如

tex　foo.bar.tex

不能用

tex　foo.bar

代替。另外即使操作系统（如 win32）允许文件名含有空格，但是 tex 的输入文件名中不能含有空格。

2.LaTeX 的使用

为了运行 LaTeX，我们可以运行 TeX 并在命令行中使用一个参数使 TeX 系统装入 LaTeX 格式文件 latex.fmt，如

tex　"&latex"　mydoc

将使 LaTeX 编译器处理输入文档 mydoc.tex。为了方便用户，MikTeX 提供了一个别名叫 latex.exe 的编译器，因此上面的命令等同于

latex　mydoc

3.dvips 的使用

dvips 工具将一个由 TeX 或其他程序生成的 dvi 文件转换成一个 PostScript 文件，通常转换结果直接被送到一个 PostScript 激光打印机。转换时可以使用驻留在打印机中的字体，也可以使用 PK 文件中以位图方式存储的字体，还可以两种方式混合使用。dvips 将自动唤醒 METAFONT 工具产生那些尚不存在的字体。

dvips 工具通常用下面的命令启动：

dvips　*options*　*dvifile*

其中 dvifile 是输入的 dvi 文件，后缀.dvi 可以省略。options 用来设置打印特性，一些主要可用的值及其含义如下：

-a：通过对.dvi 文件扫描 3 遍而不是通常的 2 遍，从而只装入实际上需要的字符，以便达到节约内存的目的。一般对于内存有限的机器有用。

-A：只打印奇数页。注意页面指的是 TeX 页面，不是顺序页面。

-b *num*：每一页的页体部分产生 num 份拷贝。

-B：只打印偶数页。注意页面指的是 TeX 页面，不是顺序页面。

-c *num*: 每一页产生 num 份拷贝，缺省值为 1。这种拷贝只是让打印机拷贝，输出的 PostScript 文件中数据并没有重复拷贝，同下面的-C 选项比较。

-C *num*: 在输出文件中将每一页的数据都复制 num 遍，因此比-c 选项的执行速度要慢。

-D num: 将分辨率的 dpi（每英寸像素数）值设置为 num，对水平分辨率和垂直分辨率都有影响。num 必须在 10 到 10 000 的范围内，如果分辨率设置得较高（如 400dpi），可能同时需要使用-Z 选项。对于位图字体，这个设置将影响系统对字体的选择；对于打印机驻留的 PostScript 字体，这个设置将影响字符的输出位置。

-e *num*: 保证每个字符实际放置位置与它在页面中忽略打印机分辨率时的初始位置相距不超过 num 个像素距离，缺省的值与分辨率有关。通过这个设置可以允许字符在预想的位置周围浮动一定距离，同时在一个单词开始时重新获得真实的开始位置，将改善单词中每个字母的输出间隔。

-f: 从标准输入设备读入.dvi 文件，并将 PostScript 文件写入标准输出设备。输入设备必须是可定位的，因此不能是管道，如果非得使用管道的话，只能写一个 shell 程序，首先将管道的输出定向到一个临时文件，然后将 dvips 的输入设备指向该文件。

-h *name*: 向 PostScript 的 "userdict"部分添加附加的头文件 name。如果 name 只是简单的-，则将从输出中去掉所有头文件。

-i: 将输入文件的每一节输出到一个单独的文件中，每个新文件的主文件名使用指定的输出文件的主文件名，后缀则使用 3 位按节排列的顺序号。

-l=*num*: 输出到第 num 页后将停止输出。

-m: 指定打印机采用人工送纸方式。

-n *num*: 最多输出 num 页，缺省值为 100000。

-o *name*: 将输出定向到 name 文件。如果没有给出 name 参数，输出文件将使用输入文件的主文件名外加.ps 后缀。

-O *offset*: 将页面的起点从缺省起点（页面左上角右下方各 1 英寸处）向右下方移动 offset 位移。offset 的格式同页面尺寸定义时的参数格式一样，使用两个坐标值并用逗号隔开，

-p=*num*: 从第 num 页开始输出。

-pp pagelist: 只有页面编号处于 pagelist 范围的页面才被输出。pagelist 可以用逗号连接多个值，也可以使用连字符表示编号起止，如 1,3-5,6-8,10。

-P *printername*: 指定输出的打印机名称 printername。

-r: 按页面顺序倒序输出，缺省为正序输出。

-S *num*: 设置每一节最多输出 num 页，通常同-L 选项一起使用。

-t *papertype*: 设置纸张类型为 papertype。

-T *offset*: 将纸张尺寸设置为 offset 大小，新的尺寸将覆盖 dvi 文件中的所有页面尺寸参数。offset 参数的格式同-O 选项中的 offset 参数。

-x *num*: 将放大倍数设置为 num/1 000，新的放大倍数将覆盖 dvi 文件中所指定的放大倍数。num 的值必须在 10 到 100 000 之间。

-X *num*: 将输出设备的水平分辨率设置为 num dpi。

-Y *num*: 将输出设备的垂直分辨率设置为 num dpi。

-Z: 为了降低 PostScript 字体下载信息的尺寸，强迫位图字体在下载前进行压缩。

9.2　emTeX 和 CCT 中文接口

9.2.1　emTeX 3.14159

emTeX 是由一位德国人 Eberhard Mattes 开发的一种 TeX 系统，可以运行在 IBM OS/2、PC-DOS 或 MS-DOS 操作系统上，1995 年推出其 3.14159 版。

1.软件包中的部件及版本

emTeX3.14159[4a]版由多个组成部件文件，这些部件包括 BibTeX 0.99c，dvidrv 1.6a，dvispell 1.0a，DVItype 3.5，fontlib 1.2b，GFtoDVI 3.0，GFtoPK 2.3，GFtype 3.1，METAFONT 2.718，MFT 2.0，MFjob 1.2a，makedot 1.2e，MakeIndex 2.12，maketcp 1.1d，PCLtoMSP 1.1a，pkedit 1.1h，pkeditpm 1.1a，PKtoGF 1.0，PKtype 2.3，PLtoTF 3.5，PXtoPK 2.2，TANGLE 4.4，TFtoPL 3.1，TeX 3.14159，TeXconv 1.2a，VFtoVP 1.2，VPtoVF 1.4，WEAVE 4.4，LaTeX 2.09<1992/03/25>，LaTeX2e <1995/06/01>，plain.tex 3.14159，plain.mf 2.71，AMSFonts 2.2。

2.TeX 目录结构和相关环境变量（OS/2 版）

emTeX 安装到硬盘上以后，自动产生如下的目录结构：

\emtex

根目录，只包含子目录，不包含普通文件。EMTEXDIR 环境变量指向该目录。

\emtex\bibinput

BibTeX 工具使用的参考文献数据库。可选环境变量 BIBINPUT 指向该目录。

\emtex\bin

可执行文件所在目录。需要将此路径加入到系统的 PATH 变量设置中。

\emtex\bmfbases

METAFONT 基本文件（宏包文件）及 mf.poo 文件（bigMETAFONT 和 mf386 使用）所在目录。可选环境变量 BMFBAS 指向该目录。

\emtex\book

OS/2 在线帮助手册目录。

\emtex\btexfmts

TeX 格式文件（如 LaTeX 宏包）和 tex.poo 文件（bigTeX 与 tex386 使用）所在目录。可选环境变量 BTEXFMT 指向该目录。

\emtex\data

各种配置文件、参数文件和数据文件所在目录。该目录可以通过 EMTEXDIR 环境变量定位。

\emtex\dist\latex

LaTeX2e 产品文件所在目录。

\emtex\dll

OS/2 动态连接库目录。该目录应当列入 config.sys 文件中的 LIBPATH 设置语句。

\emtex\doc

指导手册及工具文档文件目录。

\emtex\doc\latex209

LaTeX 2.09 文档目录。

\emtex\help

OS/2 在线帮助文件目录。该目录应当列入系统环境变量 HELP 的值中。

\emtex\htexfmts

TeX 格式文件(如 LaTeX 等宏包文件)及 htex386 工具使用的 tex.poo 文件所在的目录。可选环境变量 HTEXFMT 指向该目录。

\emtex\idxstyle

MakeIndex 使用的风格文件(style files)目录。由于 MakeIndex 工具不使用默认路径，为了使用这些风格文件，环境变量 INDEXSTYLE 必须指向该目录。

\emtex\mfbases

METAFONT 基本文件（宏包文件）及 mf.poo 文件所在目录。可选环境变量 MFBAS 指向该目录。

\emtex\mfinput

METAFONT 读入的字体源（文本）文件所在目录，应当只包含子目录，不包含普通文件。可选环境变量 MFINPUT 指向该目录。

\emtex\mfjob

MFjob 工具所用的工作(job)文件。可选环境变量 MFJOB 指向该目录。

\emtex\mftstyle

MFT 使用的风格文件所在目录。向 MFT 提供文件名时必须明确给出该目录的路径。

\emtex\remove

卸载 emtex（运行 emremove.exe）时需要的所安装的部件清单文件所在目录。

\emtex\src

所有提供源码的部件其源码文件所在目录。

\emtex\texfmts

TeX 格式文件（如 LaTeX 宏包）与 tex.poo 文件所在目录。可选的环境变量 TEXFMT 指向该目录。

\emtex\texinput

待处理的 TeX 输入文档文件及所需的宏或包所在目录，应当只包含子目录，不包含普通文件。可选环境变量 TEXINPUT 指向该目录。

\emtex\tfm

TFM 文件（TeX 字库文件）所在目录，应当只包含子目录，不包含普通文件。可选环境变量 TEXTFM 指向该目录。

\emx\bin

emx 可执行文件所在目录。需要将此目录路径加到系统的 PATH 变量中。

\emx\book

OS/2 在线手册目录。此目录应当列入 BOOKSHELF 环境变量。

\emx\dll

emx 所需要的 OS/2 动态连接库文件目录，此目录应当列入 config.sys 文件的 LIBPATH 语句

3.emTeX 系统中使用的文件的后缀及类型

这里列举的类型不包括一些通用类型（如.bmp、.bak 等）。

.aux	LaTeX 产生和使用的辅助文件
.bas	METAFONT 工具快速装入的基本文件（注意不是 BASIC 程序文件）
.bbl	BibTeX 的输出文件
.bib	BibTeX 使用的参考文献数据文件
.blg	BibTeX 脚本文件
.bst	BibTeX 使用的参考文献风格文件
.bug	错误列表文件
.cfg	LaTeX2e 配置文件
.ch	TANGLE、WEAVE 和 MFT 使用的变化记录文件
.clo	LaTeX2e 文档类选项文件
.cls	LaTeX2e 文档类文件
.cnf	dvidrv 配置文件
.dat	makefmt 使用的数据文件
.def	LaTeX2e 定义文件
.dlg	dvidrv 使用的脚本文件
.dot	dvidot 使用的参数文件
.dsb	dvispell 二进制参数文件
.dsi	dvisprep 输入文件（dvispell 参数文件）
.dtx	代码存档文件
.eng	英文文档文件
.err	dvidrv 使用的出错和警告信息文件
.fd	LaTaX 字体定义文件
.fdd	.fd 文件的源码文件
.fli	dvidrv 字库文件
.fmt	TeX 可快速装入的格式文件
.ger	德文文档文　German documentation file
.gf	METAFONT 产生的字体文件（通常也使用.#gf 后缀）
.gft	GFtype 缺省输出文件后缀
.glo	LaTeX 中\glossary 命令所产生的辅助文件
.idx	LaTeX 产生的索引项文件
.ilg	MakeIndex 脚本文件
.ind	MakeIndex 产生的索引项文件
.ins	docstrip 安装脚本文件
.ist	MakeIndex 风格文件
.lof	LaTeX 中\listoffigures 命令所产生的辅助文件

.log	TeX 和 METAFONT 使用的脚本文件
.lot	LaTeX 中\listoftables 命令所产生的辅助文件
.ltx	LaTeX 输入文件
.mac	TeXcad 宏文件
.mf	字体的源文本文件
.mfj	MFjob 工具使用的工作文件
.mft	MFT 使用的风格文件
.mtc	maketcp 工具用来产生.tcp 文件的输入文件
.opt	TeXcad 参数文件
.pic	TeXcad 产生的图形环境文件
.pk	压缩的字体文件
.pkt	PKtype 输出文件的缺省后缀
.pl	TFM 文件的可读形式
.poo	INITEX 或 INIMF 中可共享字符串库文件(pool file)
.pxl	字体文件（老式用法）
.rem	emTeX 卸载程序使用的已安装部件列表文件
.sty	LaTeX 使用的文档风格或风格选项文件

9.2.2 CCT 中文接口

1.CCT 主要组成部件

CCT是中国科学院计算数学与科学工程计算研究所张林波先生书写的一种TeX文档中使用汉字的接口软件，同时扩充了一些 TeX 命令（主要集中在汉字处理方面）。CCT 系统主要运行在中文 DOS 系统上。CCT 系统主要包括初始化程序 cctinit.exe、源文档预处理程序 cct.exe、屏幕显示驱动程序 dviscr.exe、字号定义文件 cct.dat、字库定义文件 ccfonts.def、绘图程序（pictex.exe 和 pdftobmf.exe）、几个图像图形接口程序以及几种打印机驱动程序等。1999 年又推出了 Windows 版的 dvi 文件预览工具 cctwin.exe。下面简单介绍几个排版处理程序，受篇幅所限，有关 CCT 设备驱动、图形图像处理等软件本书不作介绍。

(1) cctinit.exe

这个程序用来初始化 CCT 系统，它的作用是根据字号定义文件 cct.dat 中所定义的各个字号的大小生成 TeX 排版时需要用到的 TFM 文件以及宏文件 cchead.sty。在首次使用 CCT 系统前或每次修改了 cct.dat 中的参数后必须运行一次此程序以保证整个系统的正常运行。

(2) cct.exe

cct.exe 将 CCT 源文档转换为 TeX 源文档。用户在输入完毕 CCT 源文档后必须首先将其转换为 TeX 源文档，然后才能进行排版处理。

(3) cct.dat

CCT 的程序运行时从 cct.dat 文件中得到有关汉字大小、间距等信息。用户可以通过修改此文件来设置每种字号的尺寸。如果修改了这个文件，则需重新运行一次 cctinit.exe 程序来生成相应的.TFM 文件及 cchead.sty 文件。cct.dat 必须与 ccinit.exe 位于同一子目录中。

cct.dat 是一个普通文本文件，它的开头是一个正整数，给出可以同时使用的不同字号的个数（一般就是该文件中定义的字号数量），最大不能超过 26（也就是说一个文档中用户最多只能同时使用 26 种不同大小的字号）。接着顺序给出定义每个字号的数据，每个字号由 5 个数定义，其中前 4 个数是实数，第 5 个数为整数，它们的含义如下：第一个数表示字宽（以 pt 为单位）；第二个数表示字高（以 pt 为单位）；第三个数表示字间距与字宽之比；第四个数表示行间距与字高之比；第五个数给出用\zihao 命令选择此种字号时应该使用的参数（可以为负数）。不同数据之间需用空格或换行隔开。例如假设 cct.dat 文件中的某一行内容为

11.32　11.32　0.06　0.0　-4

则我们在 CCT 源文档中可以使用\zihao{-4}命令来使用该字号，该字号的字宽和字高均为 11.32pt，字间距为 11.32pt*0.06=0.68pt，行间距为 0pt。用户最好不要修改 cct.dat 中的内容，如果想自定义其他大小的字号，可以加在原来内容的后面。要注意 cct.dat 文件中所定义的大小不是绝对的，如果用户在 CCT 源文档中或输出结果时使用了\magnification 参数，则所有的汉字也会相应地被放大或缩小。

(4) ccfonts.def

这是一个仅供 CCT 设备驱动程序用到的一种文本文件，它定义了汉字字库的文件名。CCT 允许用户最多同时使用 26 种字体，每种字体被分为两级，分别存储在两个字库文件中。第一级字库包括区位表中第 16 区到第 55 区中的汉字，第二级字库包括第 56 区到第 87 区中的汉字。另外还有一个文件包含区位表中第 1 区到第 15 区中的符号和一个用户字库文件。ccfonts.def 文件中按下面的顺序给出这些字库文件的文件名：

用户字库
标点符号库（1~15 区）
第一种字体的一级字库文件名
第一种字体的二级字库文件名
第二种字体的一级字库文件名
第二种字体的二级字库文件名
……

前五种字体通常保留，分别指定给宋体、黑体、楷书体、仿宋体和标宋。

每个字库文件名后面还必须跟随一个比例因子，用来指明该字库中的字在输出时放大的比例因子，它的大小应当在 0.5 到 2.0 之间，否则将被系统忽略。字库文件名与比例因子之间必须用空格或换行隔开。由于驱动程序忽略这些数据之后的内容，用户可在最后一个字库文件名和比例因子之后加入自己的注解或说明。CCT 扩充了一条\ziti 命令以选取字库文件。

(5) patchdvi.exe

有时用户希望使用标准 DVI 驱动程序输出用 CCT 排版的含有汉字的 DVI 文件，例如 CCT 系统没有提供 PostScript 驱动程序，用户无法直接将排版结果转换成 PostScript 的格式。此外 CCT 的设备驱动程序目前不支持一些在西文 TeX 中广为使用的\special 命令，包括 PostScript 图形的插入、彩色打印以及 LaTeX2e 的 graphics 包中提供的缩放、旋转等功能。为此，CCT 系统中提供了一个转换程序 patchdvi.exe，它可以将含有汉字的 DVI 文件变成

标准的 DVI 文件，同时将其中的汉字转换成临时的 PK 字库，转换后的 DVI 文件可以用任何标准的 DVI 驱动程序进行显示、打印。这样用户可以根据自己的排版需要选用任何 DVI 驱动程序输出排版的结果。

(6) makepk.exe

有时打印或显示 DVI 文件时会产生形如 cannot load "???.PK"或 Cannot find subdirectory 之类的错误信息，这是因为驱动程序找不到 DVI 文件所需要的一些 PK 字库，这些 PK 字库通常由 MF 字库生成。为了方便用户生成 PK 字库，CCT 提供了一个程序 makepk.exe，其使用格式如下：

makepk [*option*] *DVIfile* [> temp.bat]

该命令生成指定的 DVI 文件 DVIfile 所用到的 PK 字库。可选参数 option 的可用值及其含义如下：

-R 给出 PK 字库的分辨率 x，如-R300、-R240:216 等。合法的分辨率定义在文件 makepk.mod 中，该文件必须与 makepk.exe 文件位于同一目录下，并且其中的定义必须与 EXTEXDIR\mfinput\etc\local.mf 文件一致，这里 EMTEXDIR 指 emTeX 根目录。缺省时使用 makepk.mod 中定义的第一种打印机的分辨率。

-M 给出放大倍数，缺省值为 1 000。

-Q 取消 makepk.exe 运行时的信息。

-A 强制生成 DVI 文件需要的所有 PK 字库。缺省时只生成所缺的 PK 字库。

-T 将产生的 TFM 文件拷贝到目录 EMTEXDIR\tfm 下。

-E 如果不使用-E 可选项，makepk.exe 并不直接生成 PK 字库，而是将生成 PK 字库的命令写到标准输出设备。用户可将这些输出定向到一个批处理文件中（如 >temp.bat），然后执行该批处理文件来产生所需的 PK 字库。如果使用了-E 可选项，则 makepk.exe 将直接执行产生 PK 字库的命令，而不是将这些命令显示出来。

使用 makepk.exe 时必须安装 emTeX 的 MF386.EXE 程序及相关文件。

(7) uncct.exe

这个程序将经过 CCT 预处理产生的文件"复原"成转换前的源文档文件。用户如果丢失了 CCT 源文件但有 CCT 预处理后产生的 TeX 文件，可以用这个程序来恢复 CCT 源文件。

(8) pscvt.exe

目前使用最广泛的 DVI 驱动程序是 dvips，它将 DVI 文件转换成 PostScript 文件。dvips 支持几种形式的\special 命令，它们被用来在排版中插入 PostScript 格式的图形文件或产生一些特殊的排版效果（对字符进行旋转、缩放等）。许多绘图软件与 TeX 的接口中都使用了 dvips 的这一功能，用户也经常用 epsf.sty 中所定义的命令在排版中插入 PostScript 图形文件。

由于 CCT 的驱动程序不支持 dvips 的\special 命令，因此，无法显示或打印出插在排版中的 PostScript 图形。为了部分地解决这个问题，CCT 系统提供了一个转换程序 pscvt.exe，它将一个包含有 dvips 的\special 命令的 DVI 文件转换成一个新的 DVI 文件，其中 dvips 的\special 命令被转换成 CCT 支持的\special{BMF=...}的形式，并自动生成相应的 BMF 文件，这样用户可以用 CCT 的设备驱动程序处理新的 DVI 文件从而得到正确的插图。

pscvt.exe 程序运行时需要调用 GhostScript 的 DOS 版程序 GS386.EXE 以及 CCT 的图

形转换程序 img2cct.exe，用户可以通过修改文件 pscvt.ini 中的-g 可选项来设定 GhostScript 的路径

(9) dviscr.exe

使用本程序可以在屏幕上显示排版。程序首先将所有字库调入内存，然后将第一页的内容显示出来。用户可以控制换页、缩放等。

(10) cctwin.exe

1999 年 7 月推出了 v1.4p10 版，用来预览 dvi 文件、支持打印、多级缩放、不同观察视角等功能。

2.CCT 排版处理过程

CCT 系统采用批处理方式。用户首先使用任一种支持汉字的文本编辑工具，按照 TeX 语法编辑带汉字的 TeX 文档，文档的扩展名使用.ctx。由于 TeX 软件不能直接处理含有汉字的 TeX 文档，用户必须使用 CCT 系统提供的一个预处理程序 cct.exe 将 CCT 源文档转换为 TeX 源文档，然后用 TeX 软件（通常配合使用 emTeX 系统）进行编译并产生一个 DVI 文件。这种 DVI 文件中含有汉字信息，同样不能直接用 emTeX 系统所配置的驱动程序来显示或打印排版结果，必须使用 CCT 系统提供的驱动程序（如 cctwin.exe）来输出排版的结果，或者用 patchdvi.exe 程序对 DVI 文件进行转换后再调用标准的 DVI 驱动程序（如 DVIPS.EXE）来进行输出。在第一次使用 CCT 系统时或 CCT 系统字体参数发生变化时还必须运行 cctinit.exe 程序。

3.CCT 扩充的与汉字处理有关的命令

除了支持标准的 TeX 命令外，CCT 系统定义了一些与汉字有关的命令。由于这些指令定义在文件 cchead.sty 中，因此用户在 CCT 源文件导言区中必须加入下面的 TeX 命令：

```
\input cchead.sty
```

或 LaTeX2e 命令

```
\usepackage{cchead}
```

来装入这些命令。下面分别说明这些命令的含义。

(1) \zihao 命令

\zihao{*num*}

用来选择汉字的大小（即字号），num 参数就是在字号定义文件 cct.dat 中每种字号的第 5 个数据。

(2) \ziti, \songti, \kaishu, \heiti, \fangsong 及\biaosong 命令

\ziti{*index*}

用来设置汉字的字体，index 参数表示所设置的字体在 ccfonts.def 文件中的序号（用不分大小写的英文字母表示）。通常\ziti{A}, \ziti{B}, \ziti{C}, \ziti{D}和\ziti{E}分别保留为宋体、黑体、楷书体、仿宋体和标宋，它们可分别使用简写版命令\songti, \kaishu, \heiti, \fangsong 及\biaosong。如果用户想要使用这五种标准字体之外的字体，则可用命令\ziti{F}, \ziti{G}等来选择。

(3) \ziju 命令

\ziju{*ratio*}

用来改变缺省的汉字字间距大小，ratio 参数表示字间距与字宽之比，缺省为 0.06。

(4) \pushziti 和\popziti 命令

这两条命令均没有参数。\pushziti 命令在输出时使得驱动程序将当前字体压入一个栈中，而\popziti 命令使得驱动程序将当前字体恢复成上一次用\pushziti 命令压入栈中的字体（栈的最大深度为 16）。

由于受 CCT 目前的处理方式的影响，如果一个可浮动体中的汉字字体与它在输出时所处位置上的其他部分的字体不同，则紧跟在浮动体后面的汉字的字体可能会被改变。汉字字体入栈和出栈命令即是为解决此问题而设置的，用户只需在一个浮动体的开始加入一条\pushziti 命令、而在最后加入一条\popziti 命令即可避免它改变所处位置上的汉字字体。

9.3 WinEdt 5.1

9.3.1 WinEdt 5.1 简介

WinEdt 是由 Aleksander Simonic 开发的大文档输入系统，尤其适合于 TeX/LaTeX 文档的可视化输入。最新版本是 2000 年 4 月推出的 5.1 版，使用 Borland Delphi 5 (Pro)编译，可以运行在 Windows 9x/NT/2000 上。作者计划当 Borland Delphi for Linux 发布后，立即将 WinEdt 移植到 Linux 上并命名为 LinEdt。

除了一般字处理软件的编辑功能外，WinEdt 支持自动换行、块和列（竖块）操作、加亮显示、拼写检查、定界符号匹配检查、转换表定义、宏定义等功能，同时支持动态数据交换(DDE)、项目管理，提供用户可定制的图形界面，支持桌面恢复功能，等等。

对于高级用户，WinEdt 额外提供了下面这些途径帮助用户定制 WinEdt 程序：

- 在文档读盘或写盘时，提供透明的转换机制，将 TeX 中特殊字符（如日耳曼语的元音变音或注音字符等）自动转换成 8 位国际编码字符中的对应符号；
- 提供一种简单的宏语言；
- 通过灵活的弹出式菜单机制，提供一种同 unix 下 emacs 编辑程序宏指令相似的“双击”(double-stroke)式快捷输入；
- 允许定义上下文相关命令；
- 提供的图形界面允许用户将一组图标同特定的命令或宏相关联；
- 能快速编译文档并产生目录，通过目录允许用户在文档内快速移动；
- 能定义引用并产生引用标签；
- 提供活动字符串(active strings)和命令完成功能。

9.3.2 用 WinEdt 进行 TeX 文档排版

虽然 WinEdit 是一个通用的文档编辑环境，但同其他类似的环境相比，它对 TeX/LaTeX 文档的输入进行了大量的改进并卓有成效。但是它并不是一个完整的 TeX 系统，而只是一个前端输入工具，后端的 TeX 编译、dvi 预览、打印或向 PostScript 文档的转换等等，仍然需要别的应用程序来处理。

经过定制可以在 WinEdt 外壳下运行多个不同的应用，这一点对于 TeX 文档的输入来说特别有用。WinEdt 允许使用菜单命令或工具条按钮激活 DOS 或 Windows 程序，因此在

定义适当的应用程序（如 TeX, LaTeX, BibTeX, Yap）之后，与书写 TeX 文档相关的大部分工作现在都可以通过点击工具按钮或选择菜单选项来轻松完成了。WinEdt 在将当前打开文档作为参数调用别的应用时，还会确保对文档的改动进行存盘，这样就保证了 TeX 文档输入与处理的一致性。WinEdt 的设计目标正是成为一个灵活的 GUI 框架，允许 TeX 用户在该框架内方便地完成诸如输入、TeX 编译、预览及 TeX 文档的拼写检查等任务流程，使作者能够将精力集中到文档的内容上来。

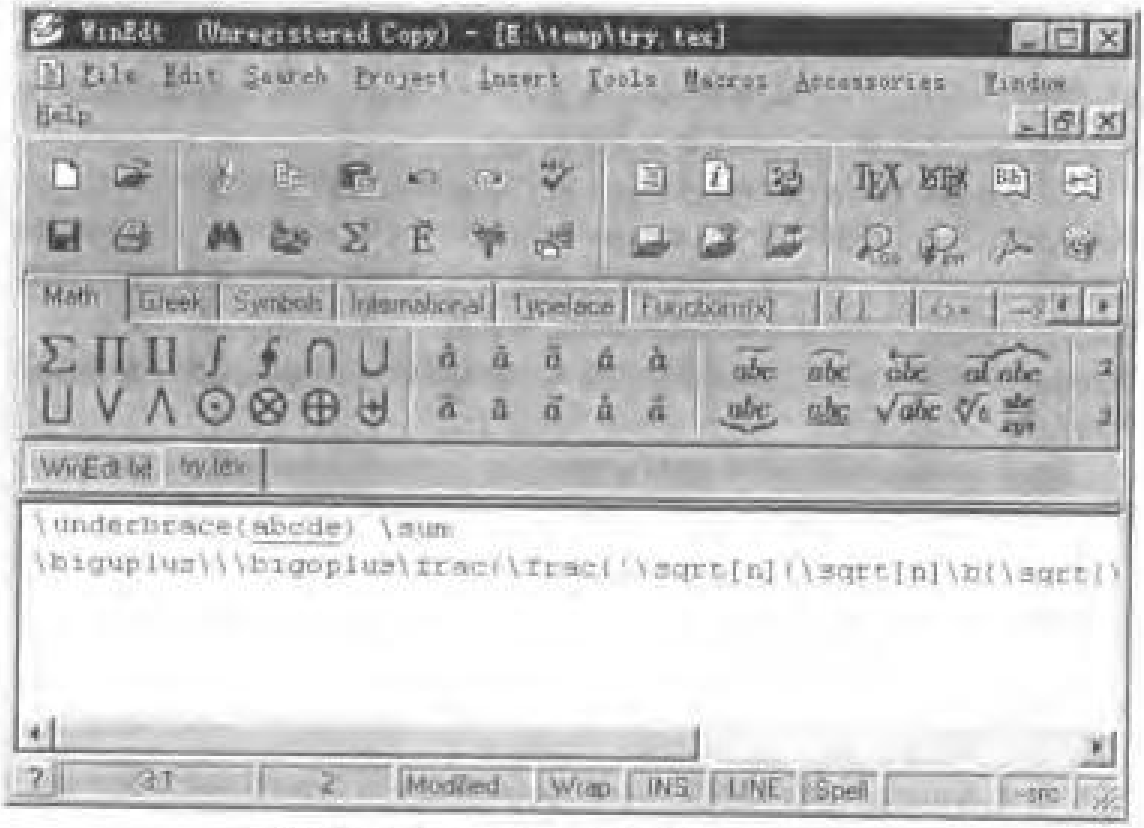


图 9.1　WinEdt5.1 输入 TeX 文档时的工作环境

缺省时 WinEdt5.1 的菜单被设置为 MikTeX 产品的前端工具风格（如图 9.1 所示）。但是用户可以通过选择 Options 菜单下面的 Menu Setup 选项，将 WinEdt 定制得符合别的 TeX 包（如 fpTeX, web2c, emTeX 等）的要求。

WinEdt 提供了一些文档（如学术论文、学位论文等）的样例，可以供 TeX 初学者进行实践。在能够编译、预览或打印这些例子之前，用户必须把所选择的配套使用的 TeX 系统正确安装。如果你想更简单、更快地进入实质性工作，MikTeX 将是很好的选择，也许正因为如此，使用 WinEdt 加上 MikTeX 系列工具的工作模式目前在国内也有为数不少的用户（尤其是 Windows 用户）。安装好 MikTeX 后，记着将\MikTeX 的 bin 子目录路径添加到系统的 PATH 变量中，否则 WinEdt 将不能正确运行 TeX 可执行文件，而且在进行诸如生成字体之类的工作时还会产生别的错误。

用户在第一次预览文档时可能要等待较长的时间，因为预览工具（如 Yap）需要花一

段时间生成 dvi 文件显示时所需的全部字体，一旦字体缓冲池建立起来以后，只是偶尔还需要生成新的.pk 文件

SRC 是一种插入 dvi 文件以允许支持 SRC 的预览工具（如 Yap0.93d 以后的版本，DVIWIN）跟踪、定位源文件的机制。WinEdt 产品中发布的“srcltx.sty”文件和“srctex.sty”文件是两个 SRC 包文件，分别在 LaTeX 文档和 TeX 文档中实现了 SRC 功能。一旦你的文档连同 SRC 包一起编译成功，则你就能够使用 WinEdt 的“DVI Search”命令在生成的 dvi 文件中预览当前行的排版结果。另外预览工具通过激活“逆向搜索”(Inverse Search)功能，可以让 WinEdt 显示所要查找的 dvi 显示部分对应的源文档位置或最接近的位置。不过这种逆向搜索的精度取决于文档的结构。另外用户必须将预览工具进行配置以便同 WinEdt 配合使用，例如对于 Yap，用户应当在 Options 菜单下的 Menu Setup 对话框中添加 Inverse Search 命令并编辑该菜单，使它对应的命令形式为：

```
C:\Program Files\WinEdt\WinEdt.exe -F "[Open(|%f|);SelLine(%l,8)]"
```

对于其他的预览工具，需要阅读它们对 Inverse Search 的支持文档。用户可以使用 WinEdt 提供的样例文档来体会 DVI 搜索工作的效果。要注意的是如果在文档的导言区中调用 DVI 搜索将没有任何结果，只有在可视文档处调用才会有用。

假如你安装的是 MikTeX1.10 或更新的版本，应当编辑 miktex.ini 文件，将 Editor 参数的设置改变为

```
Editor=C:\Program Files\WinEdt\WinEdt.exe -F "[Open(|%f|);SelLine(%l,7)]"
```

只有这样当 TeX 发生编译错误并提示时，WinEdt 才能响应用户键入的 e 命令。

9.4 EditPlus 2.01a

9.4.1 EditPlus 简介

EditPlus 是 Windows 下的 32 位文本编辑器（作者使用的是 2000 年发布的 2.01a 版），可以编辑 HTML 文档和多种高级程序设计语言（包括 CSS, PHP, ASP, Perl, C/C++, Java, Javascript 和 VBScript）源文件。EditPlus 的其他文档编辑特性主要有以下几点：

- 能够将语法（关键字）加亮显示，帮助用户尽量减少输入错误。EditPlus 已经提供 对 HTML, CSS, PHP, ASP, Perl, C/C++, Java, JavaScript 和 VBScript 等语言的语法加亮显示功能，用户还可以创建自己的语法格式文件以支持别的语言。语法格式文件是一种遵从预先指定的格式的纯文本文件，后缀名为 STX。语法格式文件的书写格式可参照一个例子，如为 JavaScript 书写的 JS.STX 文件。9.4.2 节介绍的 TeX 插件也是一个扩充的应用实例。
- 提供独具特色的 Internet 支持。EditPlus 可以在不离开编辑器的情况下无缝地浏览 HTML 文档或 Java applet，浏览窗口也可以浏览远程 Internet 站点；提供 FTP 命令帮助用户向 FTP 服务器上传本地文件，也可以直接编辑远程 FTP 服务器提供的文件；在普通文本文件中高亮度显示 URL 和 e-mail 地址，并允许用户通过单键(F8)或组合键(Ctrl + 双击鼠标)激活这类地址。Internet 特性需要 Microsoft IE 3.0 以上版本的支持，用户也可以通过 Preferences 对话框中的 Tools 组指定浏览器程序，

或者让系统装入缺省的浏览器。

- 具有文档选择工具条，用户可以在同时打开的多个文档间轻松切换，而不用使用 Window 菜单或按若干次 Ctrl+Tab 键。
- 提供 HTML 工具条，帮助用户方便快捷地插入各种 HTML 标记(tags)，同时支持 HTML 颜色、字符和对象拾取工具(picker)及表格生成器。
- 具有用户自定义能力，允许用户自定义工具。用户可以在 Preferences 对话框中增加自定义工具（下一节中有详细示例），EditPlus 提供了 10 组用户工具组，每组最多可以容纳 20 个自定义工具。自定义工具可以是程序、帮助文件或键盘击键记录文件。用户工具执行后的输出可以在 Output 窗口中捕获，用户可以在该窗口中出错的地方双击鼠标，从而自动装入产生错误的源文件并将光标定位到出错的文本行。
- 支持文档自动补全技术，该技术可以使输入工作减少。EditPlus 缺省为 Perl 和 C/C++已经提供了自动补全功能，例如在 C 程序输入过程中如果你键入了 "if"并按以下空格键，EditPlus 将输入展开为字符串

```
if ()
    {
    }
```

为了在别的文档类型中使用自动补全功能，用户应当自己书写相应的格式文件并在 Preferences 对话框的 Settings 页中指定该格式文件的路径。自动补全格式文件是后缀为 acp 的纯文本文件、其语法格式非常简单，可参照 EditPlus 提供的样例 sample.ACP 文件。

- 提供剪贴文本(cliptext)窗口，该窗口中提供可粘贴的文本库（如 HTML3.2 和 HTML4.0 标记集合），用户可以双击鼠标将其中的某些文本插入当前编辑文档。用户还可以向已有 cliptext 窗口中添加内容，或者创建自己的剪贴文本库。
- 为用户创建的新文档提供文档模板。EditPlus 为 HTML, C/C++, Perl 和 Java 文档提供了简单的模板，用户可以在 Preferences 对话框中的 Templates 页定制这些模板或添加自定义模板文件，下一节中有这项工作的例子。
- 文档尺寸可以不受限制（仅取决于系统内存），与 Windows 提供的 Notepad 不同，
- 支持多级 undo/redo 编辑操作。
- 可控制单词回绕(word wrap)。单词回绕适合于编辑很长的文本行，用户可以在 Document 菜单中打开或关闭回绕设置。
- 编辑文本可以使用 Alt 键组合鼠标拖动实现列选（类似于 WPS 中的垂直块定义），但单词回绕打开后列选功能不能使用。
- 可以显示或关闭行号，行号有助于提高 HTML 文档或源程序文件的可读性。
- 提供标尺，标尺也能提高文档的可读性，另外还可以帮助用户能很快确定光标位置。标尺也可以关闭。
- 支持 OLE 的拖放(drag&drop)编辑操作，拖放操作比剪贴板命令效率更高。在文档编辑窗口和剪贴文本窗口之间也可以进行拖放操作。
- 支持强大的搜索/替换命令，可以处理较为复杂的查找命令，同时支持在文件中查

找(Find in Files)命令，用户可以同时在多个文件中进行查找工作。用户还可以在某一文本行上设置标记，并能从文档的任何位置快速移动到标记位置。

- 支持拼写检查功能，目前只支持英语。
- 允许文档窗口撕裂(split)成多个方块，从而可以同时编辑同一文档的不同部分。
- 提供击键记录功能，便于以后重演。
- 允许用户定制 EditPlus 的所有命令的热键定义。
- 提供剪贴板监视功能，使得所有拷贝或剪切进剪贴板的内容都自动添加到当前编辑文档的末尾。这个功能适合于进行多次重复 cut&paste 操作的场合。
- 提供列标记。列标记可以标识某一特定的列，适合重视列的位置的编程语言如 FORTRAN 和 COBOL。

9.4.2 EditPlus 与 CCT emTeX 配合使用

有人开发了一套供 EditPlus 使用的 CCT emTeX 插件组（仅在 Win95 及以上版的操作系统上可用），其中包括 ctx.acp, ctx.stx, template.stx, ll.bat 等文件。如果你安装了 EditPlus，另外又安装了 CCT emTeX 和 CCT Win32 版，则利用这组插件可以将 EditPlus 与 CCT emTeX 配合使用以便方便地进行中文 TeX 文档的排版输入。这些插件可以在 http://octane.math.ustc.edu.cn/texguru/editing/处下载。

这组插件具有如下一些特点：

- 自动识别是否需要进行中文预处理；
- 自动补全\begin{}…\end{}中的后半部分；
- 编译和显示只需要按四个键；
- 查看内容的自动刷新；
- 利用彩色文字提醒你输入正确的 LaTeX 命令；
- 利用自动行号帮助定位查错。

通过如下的配置过程，可以使 EditPlus 如同编辑 HTML 文件一般编辑 emTeX 文件，不用再到 DOS 下编译和预览 TeX 文档：

(1) 把 ctx.acp, ctx.stx, template.ctx 和 template.tex 这四个文件拷贝到 EditPlus 目录中，把 ll.bat 拷贝到 emTeX 目录中。

(2) 运行 EditPlus，选择 Tool 菜单下面的 Preferences...菜单项，将会出现一个对话框。

(3) 单击对话框中左边 Categories 框中的 Files/Settings，按下对话框右边的 Add 按钮，对话框的情况将如图 9.2 所示。在 Description 栏输入文本 emTeX，在 Extension 栏输入文本 ctx;tex，并选中 Show line number 复选框。然后按下 Auto-completion 框右边的▼按钮，并在弹出的 Select File 对话框中选中 EditPlus 目录下的插件文件 ctx.acp。

(4) 单击 Categories 中的 Tools/User tools，按下 Add Tool>>按钮，在弹出的菜单中选择 Program，在 Menu text 栏输入 Compile emTeX。按下 Command 栏右边的...按钮，并在弹出的 Select File 对话框中选中 emTeX 目录下的 ll.bat 文件。单击 Argument 栏右边的▼按钮，在弹出的菜单中选择 File Name Without Extension 项，这时在 Argument 栏中自动插入$(FileNameNoExt)文本。最后选中 Close window on

exit 复选框。

(5) 在增加了 Ghat 命令后，可以同法加入命令 cctwin32.exe，这时只是把 menu text 改为 View DVI 以示区分，而 Argument 仍然选择$(FileNameNoExt)。

(6) 为了查看得到的 PostScript 文件，当系统中安装了 GhostView 时，可以同法加入命令。这时只是把 menu text 改为 View PS 以示区分，而 Argument 仍然选择 $(FileNameNoExt)，但是要在其后加上.ps，否则该命令行不通。另外注意这里用的 GhostView 文件是 gstools\gsview 下的 gsview32.exe。

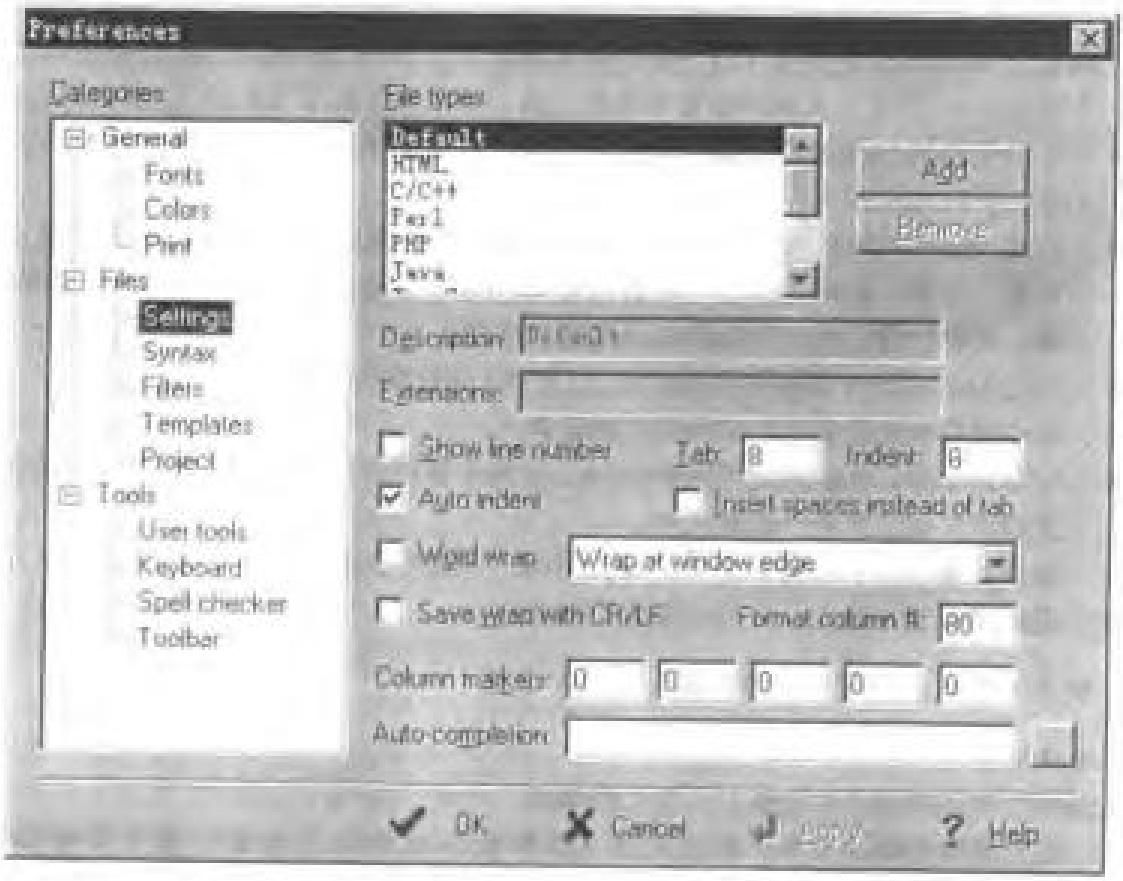


图 9.2　EditPlus2.01a 中的 Tool/Preferences...对话框

完成上述配置工作后，EditPlus 现在可以很方便地用于 TeX 排版了。EditPlus 的 File 菜单下的 New 选项在最后增加了两个新成员 CCT emTeX 和 LaTeX2e，我们任选其中的一个，马上会生成彩色 emTeX 文件模板，可供 LaTeX 新手很快上路。其中 CCT emTeX 生成的是中文 LaTeX 文件，存储时默认后缀 ctx，而 LaTeX2e 生成的是 LaTeX 文件中则不能有中文，存储默认后缀 tex。

在新生成的文件中你可以发现每行都有一个行号，它可以方便查错；所有 LaTeX 或 AMS-LaTeX 命令、环境名都用不同的彩色显示。不过如果两条系统命令之间没有任何空格，那么它们都不会变色，这是 EditPlus 的缺陷。因此在可能的情况下命令前后加上空格，这样如果它不变色，那就有错了。

如果要编译并查看一个已经输入的 TeX 文档，不需要再进入 DOS 执行那些冗长的命令，只要轻轻地按一下 Ctrl+1（必要时可按两次 Ctrl+1），然后再按 Ctrl+2 或 Ctrl+3。注意以首次查看时才需要按 Ctrl+2(3)，以后只要你不关闭查看窗口，那么只要按 Alt+Tab 进行窗口切换，其内容会自动刷新。

9.5 Scientific Notebook 3.0

9.5.1 Scientific Notebook 3.0 主要功能

Scientific Notebook 是美国 TCI Software Research 公司和 Brooks/Cole 出版公司共同开发的一种 Windows 下所见及所得的数学排版工具，1997 年发布了 3.0 版。它的主要功能有：

- 非常直观的 LaTeX 文档输入功能，使用户自然方便地输入数学公式和符号。本部分内容我们将在 9.5.2 节中作较为详细的介绍。
- 内置 Maple 引擎，可以帮助用户快速解算常用的数值计算和符号计算问题，包括代数、三角、阶乘、组合、指数函数等。另外还可以进行积分、微分、矩阵和矢量、线性代数、微分方程及统计等复杂运算。
- 利用 Maple 还可以很方便地绘制 2D 和 3D 的数学函数曲线。
- 可以进行各种度量单位之间的互换运算。
- 可自定义标记(tag)和文档风格(style，文档中使用的所有 tag 的总和)，以便于产生美观而又前后风格一致的文档。
- 提供数量庞大、分类齐全的工具及图标，环境易于使用。
- 提供拼写检查功能，并可以设置语言属性。

9.5.2 使用 Scientific Notebook 3.0 输入 LaTeX 文档

1.文档模板

Scientific Notebook 3.0 提供了三类文档模板：

Article：用来编辑常用的文章，包括科技文章模板(Scientific Article)、数学草稿纸(Math Scratchpad)、实验室报告(Lab Report)、学生家庭作业(Homework)和空白文档(Blank Document)；

Exam：用来编辑考试试卷，只有一个 Exam 模板；

Misc：用来编辑各种其他文档，包括带定理标记的空白文档(Blank with Theorem Tags)、小册子(Booklet)、信笺(Letter)、在线数学文档(On Line Math)、幻灯片(Overhead Transparencies)和个人简历(Resume)。

2.数学公式和特殊符号工具

(1) 通用符号，如图 9.3 所示。

图 9.3 Scientific Notebook 3.0 通用符号工具

(2) 第一组数学符号(Math1)，包括分式、开方、上下标、大的圆括号和方括号、求和符号和积分符号，如图 9.4 所示。另外还有一个插入各种物理常量的工具(fi lib)。

图 9.4　Scientific Notebook 3.0 第一组数学符号工具

(3) 第二组数学符号(Math2)，包括文本显示区(Display)、运算符号(Operaters)、定界符号(Brackets)、矩阵(Matrix)、函数名(Math Name)、空白、可变左右定界符号和可选分隔符号的二项式(Binomial)、标识(Label)和文本上下修饰(Decoration)，如图 9.5 所示。

图 9.5　Scientific Notebook 3.0 第二组数学符号工具

(4) 其他符号，包括大小写希腊字母、二元运算符号、二元关系符号、否定性关系符号、箭头符号、杂类符号、特殊定界符号、基本拉丁语符号、扩展拉丁语符号和一般标点符号，如图 9.6 所示。

图 9.6　Scientific Notebook 3.0 其他符号工具

Scientific Notebook 与前面介绍的软件最大的差别，在于它是真正的将 LaTeX 输入编辑工具与 LaTeX 编译系统结合在一起的软件，很大程度上可以像 Word 编排风格那样。但是它的缺点正是因为不使用单独的、通用的 LaTeX 后端处理系统，使得它对 LaTeX 所作的扩充或改动与别的 LaTeX 系统之间存在一定的兼容问题。另外 Scientific Notebook 缺乏对汉字处理能力的支持。

附录 LaTeX 命令一览表

本附录列出 LaTeX 中的绝大部分命令及其简单描述，其中有些是 TeX 标准命令，在此作者不作特别区分。书中第五章所介绍的 AMS LaTeX 扩充的命令并没有在此列出。每条命令解释内容后的数字表示该命令在本书中的页码，对于在不同上下文中有几种不同语义的命令，我们将标示出多个页码，分别对应多种语义各自出现的页码。少数命令在本书中没有介绍，所以就没有页码。

!` ¡ 符号(p.30)。

?` ¿ 符号(p.30)。

$ 成对出现，表示数学环境。$...$相当于\(...\) 。比照\(命令和\)命令(p.47)。

% 注释命令，表示本行后面的内容均为注释，不为文档输出内容(p.5)。

_{*text*} 将文本 text 以下标的形式输出(p.53)。

^{*text*} 将文本 text 以上标的形式输出(p.53)。

\␣ 与点句号之后原始空白同样大小的空格(p.57)。

\, 留一点小小的空白(p.57)。

\! 用于数学模式，产生 1/6 空铅大小的负空格（即前面的空白变小），如 *xx*\!*x* 产生 *xxx* 排版效果(p.57)。

\" 产生日耳曼语系中的元音变音符号，如 ö 符号用\"o 产生等(p.30)。

\# 输出一个英镑符号(#)(p.6)。

\$ 输出一个美元符号($)(p.6)。

\% 输出一个百分号(%)(p.6)。

\& 输出一个&符号(p.6)。

\' 在制表位环境中，将该命令左右两边的文本按照当前制表位靠紧；其中的间隔为 \tabbingsep；在其他环境中产生重音符号(p.22, 30)。

\(开始进入数学模式，同\begin{math}命令或$命令(p.47)。

\) 结束数学模式，同\end{math}命令或$命令(p.47)。

* 任意乘法符号，允许换行符号出现。

\+ 用于制表位环境，使本行开头右移到下一个制表位(p.21)。

\- 在制表位环境中使本行开头左移到前一个制表位；在其他环境中表示允许连字符出现即段字的位置(p.21)。

\. 在字母上方放置一个圆点的读音符号(p.30)。

\/ 插入斜体字体调整空格。

\: 用于数学模式，产生中等大小（2/9 空铅）的空格(p.82)。

\; 用于数学模式，产生较大（5/18 空铅）的空格(p.82)。

\< 用于制表位环境，将文本放置在当前左边界的左边(p.21)。

\= 在制表位环境中放置一个制表位；在其他环境中产生长音注音符号（如 ō ）(p.21, 30)。

\> 在制表位环境中表示向前分栏符；在数学环境中产生中等大小（2/9 空铅）的空格(p.21)。

\@ 声明后跟的圆点是句号(p.29)。

\[同\begin{displaymath}命令或 $...$命令(p.47)。

\\ [*extra-space*] 终止一行，extra-space 指明下一行开始前预留多少垂直距离（可以为负值）(p.14)。

*[*extra-space*] 终止一行，但不允许换页。extra-space 参数含义同\\命令(p.14)。

\] 同\end{displaymath}命令或$$命令(p.47)。

\^ 产生抑扬符号（如 ô）(p.30)。

_ 下划线(p.6)。

\` 在制表位环境中将直到\\为止的所有后续文本移到右边界处（即改为右对齐）；在其他环境中产生低音发音符号（如 ò）(p.22, 30)。

\{ 打印左花括号(p.6)。

\| 数学模式下的 ‖ 符号(p.57)。

\} 打印右花括号(p.6)。

\~ 产生波浪型发音符号（如 ñ）(p.30)。

\a' 用于制表位环境，代替\'命令来产生锐音重音符号（如 ó）(p.22)。

\a` 用于制表位环境，代替\`命令来产生低音发音符号（如 ò）(p.22)。

\a= 用于制表位环境，代替\=命令来产生长音注音符号（如 ō ）(p.21)。

\aa 输出 å 符号(p.30)。

\AA 输出Å符号(p.30)。

\acute 用于数学模式，产生锐音重音符号（如 ó）(p. 48)。

\addcontentsline{*toc*}{*section*}{*name*} 向 toc 参数指定的.toc 文件中添加\contentsline {section}{name}命令。

\address{*text*} 用于文本模式，声明返回地址。

\addtocontents{*toc*}{*text*} 向 toc 参数指定的.toc 文件中添加文本 text 。

\addtocounter{*counter*}{*value*} 使计数器 counter 的值增加 value 大小(p.17)。

\addtolength{*gnat*}{*length*} 使长度命令\gnat 所代表的长度增加 length 大小。参见\setlength、\newlength 以及\settowidth 命令(p.25)。

\ae 输出 æ 符号(p.30)。

\AE 输出 Æ 符号(p.30)。

\aleph 数学模式下的ℵ符号(p.52)。

\alph{*counter*} 以小写英文字母打印计数器 counter 的值(p.17)。

\Alph{*counter*} 以大写英文字母打印计数器 counter 的值(p.17)。

\alpha 数学模式下的 α 符号(p.3)。

\amalg 数学模式下的∐符号(p.50)。

\and 在\maketitle 命令中分隔多个作者名(p.9)。

\angle 数学模式下的∠符号(p.52)。

\appendix 文档附录开始命令(p.26)。

\approx 数学模式下的≈符号(p.49)。

\arabic{*counter*} 以阿拉伯数字的形式打印计数器 counter 的值，如 1, 2, 等等(p.17)。

\arccos 用于数学模式，代表 arccos 函数名字符串(p.53)。

\arcsin 用于数学模式，代表 arcsin 函数名字符串(p.53)。

\arctan 用于数学模式，代表 arctan 函数名字符串(p.53)。

\arg 用于数学模式，代表 arg 函数名字符串(p.53)。

\arraycolsep 用于矩阵环境中，指定矩阵中两列之间的间隔大小(p.88)。

\arrayrulewidth 用于表格或矩阵环境中，代表由\hline 或\vline 命令产生的标尺宽度(p.101)。

\arraystretch 用于表格或矩阵环境中，代表行间间隔的放大倍数因子(p.101)。

\ast 数学模式下的*符号(p.50)。

\asymp 数学模式下的≍符号(p.49)。

\AtBeginDocument{*code*} 用于类或包文件中，声明将code代码参数暂时内部保存，等到LaTeX执行\begin{document}命令时再执行(p.108)。

\AtBeginDvi{*specials*} 用于类或包文件，将specials参数保存在一个注册盒子里并写入.dvi文件中文档首页输出开始处(p.116)。

\AtEndDocument{*code*} 用于类或包文件中，声明将code代码参数暂时内部保存，等到LaTeX执行\end{document}命令时再执行(p.108)。

\AtEndOfClass{*code*} 主要用于\DeclareOption或\DeclareOption*命令的参数中间，将code代码暂时内部保存，等到当前类处理完毕后再执行code代码(p.116)。

\AtEndOfPackage{*code*} 要用于\DeclareOption或\DeclareOption*命令的参数中间，将code代码暂时内部保存，等到当前包处理完毕后再执行code代码(p.116)。

\atop 产生二项组合式(p.56)。

\author{*names*} 声明\maketitle 命令中的作者姓名(p.9)。

\b 下画线注音符号(p.30)。

\backslash 数学模式下的\（反斜线）符号(p.6)。

\bar 用于数学模式，在字母上面放置一个长音标注(p.48)。

\baselineskip 段落中一行文字的底部离下一行文字底部的距离(p.16)。

\baselinestretch 每当执行改变字体尺寸的命令时，作用于\baselineskip 的缩放比例因子。

\begin{*environment*} 总是同\end{environment}命令成对出现，用以定义环境。下面列举了各种环境的定义命令(p.18)。

\begin{abstract} 开始产生文档摘要的环境。

\begin{array}{*lrc*} 开始一个三列的矩阵环境，其中三列的对齐方式分别为左对齐、居中、右对齐（由 lrc 指定）。每一列的内容间用&分隔，列的末尾用\\结束。l,r,c 中间的@{text}参数将放置在相应两列文本的中间(p.57)。

\begin{center} 开始每一行文字都居中对齐的环境，每一行用\\结尾(p.19)。

\begin{description} 开始带标识的列表环境，每一个列表的项用\item[*keyword*]命令给

出(p.31)

\begin{displaymath} 设置数学模式，同\[或$...$命令(p.47)。

\begin{document} 实际文档文本开始，每个文档必须要有这条命令(p.6)。

\begin{enumerate} 开始带编号的列表环境(p.30)。

\begin{eqnarray} 开始可以容纳多个公式的数学模式。公式间用\\或*命令分隔；\nonumber 命令用于省略特定公式的编号(p.47)。

\begin{eqnarray*} 除了公式不进行编号以外，同\begin{eqnarray}命令(p.48)。

\begin{equation} 开始一个 displaymath 环境并添加一个公式编号(p.47)。

\begin{figure}[*placement specifier*] 开始一个浮动(floating)环境，其位置可以由可选参数 placement specifier 指定，有关 placement specifier 参数的意义参见本附录后面的 positions 条目(p.102)。

\begin{figure*}[*placement specifier*] 开始一个两列宽的浮动图形环境。

\begin{flushleft} 在开始的环境中，文本保持右边粗糙（即同左边注保持平齐），行间用\\命令分隔。参见\raggedright 命令(p.19)。

\begin{flushright} 在开始的环境中，文本保持左边粗糙（即同右边注保持平齐），行间用\\命令分隔。参见\raggedleft 命令(p.19)。

\begin{itemize} 开始带圆点(●)标识的列表环境，每一个列表项用\item 命令给出(p.30)。

\begin{list}{*default-labeling*}{*spacing*} 开始一个通用列表环境(p.33)。

\begin{lrbox}{*cmd*} 文本盒子环境，cmd 为一个代表盒子的命令(p.42)。

\begin{math} 在文本模式下开始一个数学环境，同$或\(命令(p.47)。

\begin{minipage}[*position*]{*width*} 开始一个宽度为 width 的微型页面环境。文本的放置位置取决于 position 参数，有关 position 参数的意义参见本附录后面的 positions 条目(p.20)。

\begin{picture}(*x*,*y*) 开始一个图形环境，其右上角坐标为(x, y)。坐标单位用 \unitlength 命令设置(p.89)。

\begin{quotation} 开始较宽边注、正常段落缩进量、保持上下边距的环境(p.19)。

\begin{quote} 开始较宽边注、段落不缩进、保持上下边距的环境(p.19)。

\begin{tabbing} 开始制表位环境(p.20)。

\begin{table}[*placement specifier*] 开始一个浮动表环境，其位置可由可选参数 placement specifier 指定，有关位置参数参见本附录后面的 positions 条目(p.102)。

\begin{table*}[*placement specifier*] 开始一个两列的浮动表环境。

\begin{tabular}[*pos*]{*cols*} 既可用于数学模式又可用于非数学模式，开始一个表格环境。pos 表示表格相对于外部文本行基线的位置：t 表示表格顶部与当前外部文本行的基线重合；取 b 表示表格底部与外部文本行的基线重合；不使用或缺省时表格按照外部文本行的基线垂直居中。cols 包含列文字定位命令：如 r, l, c, @{...}, p{length}（p 命令参见 positions 条目）；| 用于产生列间的竖线；*{7}{rll} 将大括号中的内容重复 7 次(p.98)。

\begin{thebibliography}{*width*} 参考文献列表环境，width 指定编号所占用的最大宽度

值(p.44)。

\begin{theorem} 参见\newtheorem 命令。

\begin{titlepage} 开始首页环境，首页无页码，其后面的一页页码则成为“1”。

\begin{trivlist} 开始简洁列表环境(p.36)。

\begin{verbatim} 开始一个完全按输入字面输出的环境，如回车将引起换行等等，通常使用打字机字体(p.20)。

\begin{verse} 开始一个较宽边注、无段落缩进、右边粗糙的诗歌环境(p.19)。

\beta 数学环境中的 β 符号(p.48)。

\bezier{*number*} $(x_1, y_1)(x_2, y_2)(x_3, y_3)$ 在(x_1, y_1)点到(x_3, y_3)点之间绘制一条bézier曲线，(x_2, y_2)为曲线控制点。number参数表示曲线所经过的路段数目(p.95)。

\bf 切换至粗体字。

\bfseries 将文本字体加黑。同\textbf 命令(p.37)。

\bibitem{*marker*} 产生一个参考文献列表项，其中 marker 用于标识该项参考文献，文档中可以根据 marker 引用该参考文献(p.44)。

\bibliography{*file*} 在当前文档位置插入由 file 参数指定的参考文献文件(.bib)。

\bibliographystyle{*style*} 同\documentstyle 命令类似，用于指定参考文献的格式。

\big, \Big, \bigg, \Bigg 作为命令前缀，可以得到尺寸越来越大的定界符号(p.51)。

\bigcap 数学模式下的$\bigcap$符号(p.50)。

\bigcirc 数学模式下的$\bigcirc$符号(p.50)。

\bigcup 数学模式下的$\bigcup$符号(p.50)。

\bigodot 数学模式下的$\bigodot$符号(p.50)。

\bigoplus 数学模式下的$\bigoplus$符号(p.50)。

\bigotimes 数学模式下的$\bigotimes$符号(p.50)。

\bigskip 标准的“big”型（较大型）纵向间隔(p.40)。

\bigskipamount \bigskip 命令的缺省尺寸(p.41)。

\bigsqcup 数学模式下的$\bigsqcup$符号(p.50)。

\bigtriangledown 数学模式下的$\bigtriangledown$符号(p.50)。

\bigtriangleup 数学模式下的$\bigtriangleup$符号(p.50)。

\biguplus 数学模式下的$\biguplus$符号(p.50)。

\bigvee 数学模式下的$\bigvee$符号(p.50)。

\bigwedge 数学模式下的$\bigwedge$符号(p.50)。

\bmod 用于数学模式，表示二元求模运算表达式(m mod n) (p.71)。

\boldmath 在数学模式外部使用，使其后面数学模式中的斜体字和数学符号变为粗体(p.54)。

\bot 数学模式下的$\bot$符号(p.52)。

\botfigrule 在页面底部浮动图表之前被执行的命令(p.105)。

\bottomfraction 每页中底部可浮动体占据整页高度的最大比例(p.104)。

\bowtie 数学模式下的$\bowtie$符号(p.49)。

\Box 数学模式下的□符号(p.52)。

\breve 用于数学模式，产生短弱发音符号（如 ă）(p.48)。

\bullet 数学模式下的•符号(p.50)。

\c 产生变音符号（如 ç）(p.30)。

\cal 用于数学模式，产生书法字母（如𝒜字母）。

\cap 数学模式下的∩符号(p.50)。

\caption[*short caption*]{*caption text*} 为图形或表格环境产生带编号的标题（文字由 caption text 参数给出）。如果可选参数 short caption 与 caption text 内容不同，则代表该图表在相应图表列表中的项目使用 short caption 所指定的文字(p.103)。

\cc{*text*} 用于书信(letter)文档类型、text 参数指明同时接收此信的收件人。

\cdot 数学模式下的•符号(p.50)。

\cdots 用于数学模式，产生在当前行中垂直居中的三个圆点…。比照\ldots 命令(p.52)。

\centering 声明其后的文本居中。比照\begin{center}命令(p.36)。

\chapter[*toctitle*]{*text*} 开始新的、自动题注和编号的章次，标题由 text 给出。如果可选参数 toctitle 与 text 不同，则代表该章节在目录表中的条目名称使用 toctitle 所指定的文字(p.26)。

\chapter*{*title*} 与\chapter{title}命令相似，只是不带章次编号，而且也不能在目录中使用不同的文字表示该章次。

\check 用于数学模式，产生形如 š 的发音符号(p.50)。

\CheckCommand {*cmd*} [*num*] [*default*] {*definition*} 用于类和包文件，检查cmd命令的当前定义是否为definition参数所给出的那样，如果检查的结果是不一样，则将导致出错(p.119)。

\chi 数学模式下的χ符号(p.48)。

\choose 产生二项组合式(p.56)。

\circ 数学模式下的∘符号(p.50)。

\circle{*d*} 用于图形环境，可作为\put 命令的合法参数绘制一个直径为 d 的圆形(p.94)。

\circle*{*d*} 同\circle 命令相似，只是绘制的是实心圆(p.94)。

\cite[*subcit*]{*marker*} 产生一个用方括号括起来的交叉引用，该引用指向由 ref 参数确定的相应\bibitem{marker}命令所建立的参考文献条目。可选参数 subcit 可以在相应的参考文献条目中添加子引用(p.44)。

\ClassError {*class-name*} {*error-text*} {*help-text*} 用于单独的类或包文件，当类文件 class-name出错时负责向用户报告或提供相关错误信息(p.118)。

\ClassInfo {*class-name*} {*info-text*} 同\ClassWarning命令相似，只是将info-text信息及警告发生行行号不显示而是写入log文件(p.119)。

\ClassWarning {*class-name*} {*warning-text*} 用于单独的类或包文件，当类文件 class-name出错时在屏幕上产生warning-text警告信息及警告发生处的行号，但没有帮助信息(p.119)。

\ClassWarningNoLine {*class-name*} {*warning-text*} 同\ClassWarning命令相似，只是不显示警告发生处的行号(p.119)。

\cleardoublepage 强制指定下一页为处于右边的奇数页码页(p.104)。
\clearpage 立即结束当前页，并将本页未排版的图形或表格放到后面无文本的纯可浮动页中(p.104)。
\cline{*n-m*} 用于矩阵或表格环境，绘制一条从第 n 列到第 m 列的水平直线(p.100)。
\closing{*text*} 结束书信类型的文档。
\clubsuit 数学模式下的♣符号(p.52)。
\color{*col_spec*} 将此后输出的文本的颜色切换到 col_spec 颜色(p.125)。
\colorbox *col_spec*{*text*} 将 text 参数所代表的文本输出到一个盒子中，盒子的背景颜色为 col_spec(p.126)。
\columnsep 指定双栏文档中分栏间距离。
\columnseprule 双栏页面中分栏间标尺宽度。
\columnwidth 当前分栏的宽度。对于单栏(不分栏)文档，其值与\textwidth 相等(p.105)。
\cong 数学模式下的≅符号(p.49)。
\coprod 数学模式下的∐符号(p.50)。
\copyright ©符号(p.52)。
\cos 用于数学模式，代表 cos 函数名字符串(p.53)。
\cosh 用于数学模式，代表 cosh 函数名字符串(p.53)。
\cot 用于数学模式，代表 cot 函数名字符串(p.53)。
\coth 用于数学模式，代表 coth 函数名字符串(p.53)。
\csc 用于数学模式，代表 csc 函数名字符串(p.53)。
\cup 数学模式下的∪符号(p.50)。
\CurrentOption 用于类或包文件，代表当前使用的装入可选项(p.111)。
\d 产生下加圆点的发音符号(p.30)。
\dag 非数学符号†(p.52)。
\dagger 数学模式下的†符号(p.50)。
\dashbox{*dwid*}(*width*,*height*)[*pos*]{*text*} 用于图形环境，在文本 text 四周绘制点画线方框。点画线宽度为 dwid 个长度单位；方框的宽和高分别为 width 和 height；text 放置位置可由可选参数 pos 指定，有关 pos 的意义参见 position 条目(p.91)。
\dashv 数学模式下的⊣符号(p.49)。
\date{*text*} 将文档的日期设置成 text 参数所表示的日期。如果文档中没有使用此命令，LaTeX 将把当前日期作为文档的日期(p.13)。
\day 当前日期天的数值。
\dblfigrule 用于双栏文档，含义同\topfigrule 命令(p.105)。
\dblfloatpagefraction 双栏浮动页中最少要被可浮动文档占据的页面比例(p.104)。
\dblfloatsep 双栏浮动页中顶部或底部可浮动文档间的间距(p.105)。
\dbltextfloatsep 双栏页面中顶部或底部可浮动文档与页面中文本文档之间的间距(p.105)。
\dbltopfraction 双栏页面中顶部可用于可浮动文档部分占整个页面的最大比例(p.104)。
\ddag 非数学符号‡(p.52)。
\ddagger 数学模式下的‡符号(p.50)。

\ddot 用于数学模式，在字母上放置分音符号(p.48)。

\ddots 用于数学模式，产生对角线方向的省略号（符号⋱）(p.52)。

\DeclareOption{*option*}{*code*} 类和包文件中声明被装入时可用的选项(p.110)。

\DeclareOption*{*code*} 类和包文件中的代码声明命令。如果用户使用了类或包未定义的可选项，则执行code代码(p.110)。

\definecolor{*name*}{*model*}{*specs*} 给某种颜色命名为 name，颜色的编码方式为 model，编码值为 specs(p.125)。

\deg 用于数学模式，代表 deg 函数名字符串(p.53)。

\delta 数学模式下的δ符号(p.48)。

\Delta 数学模式下的Δ符号(p.49)。

\depth 文本字符深度(p.25)。

\det 用于数学模式，代表 det 函数名字符串(p.53)。

\diamond 数学模式下的$\diamond$符号(p.50)。

\Diamond 数学模式下的$\Diamond$符号(p.52)。

\diamondsuit 数学模式下的$\diamondsuit$符号(p.52)。

\Digamma 数学模式下的$\digamma$符号(p.49)。

\dim 用于数学模式，代表 dim 函数名字符串(p.53)。

\displaystyle 用于数学模式，将输出字号切换到标准数学字体大小(p.60)。

\div 数学模式下的÷符号(p.50)。

\documentclass[*options*]{*class*} 指明所要创建的文档类别为 class(p.8)。

\documentstyle[*substy*]{*sty*} 指定文档类型为 sty，同时可以确定缺省的字体、标题等。sty 可以取 article, book, letter, report, slides 等。另外可选参数 substy 可以用来指定子文档类型，可以取 11pt, 12pt, acm, draft, fleqn, leqno, twocolumn, twoside 等。

\dot 用于数学模式，在字母上方加一圆点(p.48)。

\doteq 数学模式下的≐符号(p.49)。

\dotfill 产生一串到行尾的橡皮长度的点（不是空格）(p.25)。

\dots 数学模式下的…符号(p.52)。

\doublerulesep 表格或矩阵环境中用||产生的两条竖线分隔符号之间的间隔(p.101)。

\downarrow 数学模式下的$\downarrow$符号(p.51)。

\Downarrow 数学模式下的$\Downarrow$符号(p.51)。

\ell 数学模式下的符号ℓ(p.52)。

\em 使强调(emphasis)环境中的文字从罗马字体(roman)到斜体(italic)之间来回切换。

\emph{*text*} 将文本 text 的字型设置为强调字型（\textit 和\textrm 间切换）(p.28)。

\emptyset 数学模式下的$\emptyset$符号(p.52)。

\encl{*text*} 用于书信文档，声明一系列随信附件。

\end{*environment*} 结束由\begin{environment}命令开始的环境。参见\begin{environment}命令(p.18)。

\enlargethispage{*size*} 按照指定的尺寸 size 增大当前页面的\textheight 参数(p.16)。

\enlargethispage*{*size*} 尽量将当前页面中的内容往一块儿压缩，通常与一条显式的换页命令\pagebreak 一起使用(p.16)。

\epsilon 数学模式下的ϵ符号(p.48)。

\equiv 数学模式下的≡符号(p.49)。

\eta 数学模式下的η符号(p.48)。

\evensidemargin 双面印刷时偶数页页面左边界到文本正常左边注的距离。

\ExecuteOptions {*options-list*} 用于类或包文件。对于options-list参数中的每一个可选项，按照顺序依次执行一下\ds@option命令，如果这条命令没有定义，则该可选项将被忽略(p.117)。

\exists 数学模式下的∃符号(p.52)。

\exp 用于数学模式，代表 exp 函数名字符串（以 e 为底的指数函数）(p.53)。

\extracolsep{*colsep*} 用于表格环境中的@表达式,将后面所有列间间距在原有标准间隔的基础上增加 colsep 大小(p.100)。

\fbox{*text*} 给 text 文档四周加一边框(p.41)。

\fboxrule \fbox 命令或\framebox 命令产生的边框的线宽(p.41)。

\fboxsep \fbox 命令或\framebox 命令产生的边框与其中的文本之间的间距(p.41)。

\fcolorbox *col_spec1 col_spec2* {*text*} 同\colorbox 命令的功能相似，只是盒子的背景颜色为 col_spec2，而且带一个颜色为 col_spec1 的边框。两种颜色参数要么都是预定义的命名颜色，要么使用同一种编码模式，若为后者编码模式只需要书写一次(p.126)。

\fill 一种可以伸长到任意长度的橡皮长度，通常用于对文本进行特殊调整(p.100)。

\flat 数学模式下的♭符号(p.52)。

\floatpagefraction 浮动页面中可浮动文档占整页的最小比例(p.104)。

\floatsep 可浮动文档在下一页面顶部或底部排版时相互之间的间距(p.104)。

\flushbottom 使得页面向下拉伸到\textheight 所代表的位置。

\fnsymbol{*counter*} 以某一组脚注符号(footnote symbols)打印计数器 counter 的值。counter 的值必须小于 10(p.17)。

\fontencoding{*enc*} 选定字体编码方法，有效的方法有 OT1 和 T1 两种(p.38)。

\fontfamily{*family*} 选择字体家族 family(p.38)。

\fontseries{*series*} 选择字体修饰 series(p.38)。

\fontshape{*shape*} 选择字体形状 shape(p. 39)。

\fontsize{*size*}{*skip*} 选择字号大小。size 指定要切换成的字号大小，skip 相当于\baselineskip 命令的参数(p.39)。

\footheight 页面底部显示页码的方形区域的高度。

\footnote[*number*]{*text*} 为文本 text 创建脚注(p.27)。

\footnotemark 将脚注编号插入文档当前位置(p.27)。

\footnotesep 脚注开始处跳过的垂直空白的高度。

\footnotesize 将字体大小切换成脚注字体大小(p.38)。

\footnotetext[*number*]{*text*} 将\footnotemark 命令指定的脚注的文字设置为 text(p.27)。

\footskip 页面中最后一行文本的底部距页面最下面区域底部的距离(p.43)。

\forall 数学模式下的∀符号(p.52)。

\frac{*text1*}{*text2*} 产生一个分式,分子为 text1,分母为 text2(p.55)。

\frame{*element*} 绘制一个矩形边框，内部再绘制任何类型的图形元素(p.95)。

\framebox[*width*][*position*]{*text*} 同\makebox 命令的作用完全相同，只是在创建的盒子四周放一个方框。同时这条命令创建一个标尺，标尺的厚度由\fboxrule 指定，标尺离盒子中内容的距离由\fboxsep 指定(p.41)。

\framebox(*width,height*)[*pos*]{*text*} 用于图形环境，在 text 文档周围绘制一个矩形，矩形的宽度和高度分别为 width 和 height。文本在矩形中的放置位置可由 pos 参数指定，有关位置问题参见本附录中的 positions 条目(p.90)。

\frenchspacing 告诉 LaTeX 在句点后面不要放比一般字符后面更多的空格，许多非英语文档中这个命令很重要(p.29)。

\frown 数学模式下的⌢符号(p.49)。

\fussy 有关换行断字的缺省声明。可与\sloppy 命令比较(p.15)。

\gamma 数学模式下的γ符号(p.48)。

\Gamma 数学模式下的 Γ 符号(p.49)。

\gcd 用于数学模式，代表 gcd 函数名字符串（最大公约数）(p.53)。

\ge 数学模式下的≥符号(p.49)。

\geq 数学模式下的≥符号(p.49)。

\gets 数学模式下的←符号(p.51)。

\gg 数学模式下的≫符号(p.49)。

\glossary{*text*} 另外通过一条\glossaryentry 命令，可以将 text 附加到.glo 文件的末尾。

\glossaryentry{*text*}{*ref*} 交叉引用 ref 中出现的\glossary{text}命令将被写入.glo 文件。

\grave 用于数学模式，产生低音发音符号(p.48)。

\H 输出一个匈牙利长元音变音（如 ő）(p.30)。

\hat 用于数学模式，产生一个抑扬符号（如 ô）(p.48)。

\hbar 数学模式下的符号ħ(p.52)。

\headheight 页面顶部容纳栏外标题(running head)的方形区域的高度(p.43)。

\headsep 标题底部距文本顶部的垂直距离(p.43)。

\heartsuit 数学模式下的♡符号(p.52)。

\height 文本字符高度(p.25)。

\hfill 相当于\hspace{\fill}命令。比照\fill 命令(p.25)。

\hline 用于表格环境或矩阵环境，绘制一条跨越所有列的水平直线(p.99)。

\hom 用于数学模式，代表 hom 函数名字符串(p.53)。

\hookleftarrow 数学模式下的↩符号(p.51)。

\hookrightarrow 数学模式下的↪符号(p.51)。

\hrulefill 用横线填充水平空白(p.25)。

\hspace{*length*} 保留长度为 length 的水平空白(p.40)。

\hspace*{*length*} 同\hspace{length}命令相似，但空白处于行首或行尾时也不被删除或

忽略(p.40)

\huge 切换到相当大的字体(p.38)

\Huge 切换到比\huge 还要大的字体(p.38)。

\hyphenation{*word list*} 声明 word list 文本中的每个单词允许用连字符断开换行的位置(p.15)

\i 字符 ı (p.30)。

\iff 数学模式下的⇔字符(p.51)。

\IfFileExists {*file-name*} {*true*} {*false*} 用于类或包文件，检测文件file-name是否存在。如果存在则执行true参数给出的代码；如果不存在则执行false参数给出的代码这条命令并不自动装入待查文件(p.118)。

\Im 数学模式下的ℑ字符(p.52)。

\imath 数学模式下的ı字符(p.52)

\in 数学模式下的∈字符(p.49)。

\include{*filename*} 在当前位置插入 filename 文件中的内容(p.8)。

\includegraphics [*llx, lly*] [*urx, ury*] {*file_name*} 插入图像 file_name。可选参数定义了图像的剪裁范围。graphics 包中命令(p.122)。

\includegraphics [*key=value,...*]{*file_name*} 插入图像 file_name，可选参数定义了图像的加工方式。graphicx 包中命令(p.123)。

\includeonly{*filename1, filename2, ...*} 限制\include{filename}命令中的 filename 参数只能是 filename1 或 filename2 等(p.8)。

\indent 产生一块水平空白区域，其宽度等于段落的缩进距离值(p.16)。

\index{*key*} 通过另外的\indexentry 命令可以将 key 文本添加到.idx 索引文件的末尾(p.45)。

\indexentry{*text*}{*ref*} 对 ref 交叉引用中出现的\index{text}，将 text 写入.idx 文件

\indexspace 在以新字母开头的第一个索引项之前留空。

\inf 用于数学模式，代表 inf 函数名字符串(p.53)。

\infty 数学模式下的∞字符(p.52)

\input{*filename*} 在当前文档位置插入 filename.tex 文件内容(p.8)。

\InputIfFileExists {*file-name*} {*true*} {*false*} 用于类或包文件。如果file-name文件存在，则执行true参数给出的代码并马上装入该文件；如果file-name文件不存在，则执行false参数给出的代码(p.118)。

\int 数学模式下的∫字符(p.50)。

\intextsep 文本中间的可浮动体上下放置的垂直空白(p.105)。

\iota 数学模式下的ι字符(p.48)。

\it 切换到 italic 字体。

\item[*option*] *text* 用于各种列表环境，表明 text 文本是一个列表项。option 表示列表环境的类型(p.31)。

\itemindent 列表项中标识之前额外的缩进量，缺省为 0mm(p.33)。

\itemsep 相邻列表项之间的垂直间隔(p.33)

\itshape 将文本字型切换为斜体(italic) 同\textit 命令(p.37)。

\l 字符 ł(p.30)

\jmath 数学模式下的ȷ字符(p.52)。

\Join 数学模式下的⋈字符(p.49)。

\kappa 数学模式下的κ字符(p.48)

\ker 用于数学模式，代表 ker 函数名字符串(p.53)。

\kill 用于制表位环境，定义一个“假想”行以便不用输出文字就可以设置制表位(p.22)。

\L 字符Ł(p.30)。

\label{*key*} 提供一个交叉引用点，可以通过\ref{key}命令或\pageref{key}命令访问(p.18)。

\labelenumi enumerate 环境在列表嵌套中处于第一层时使用的列表项标识(p.32)。

\labelenumii enumerate 环境在列表嵌套中处于第二层时使用的列表项标识(p.32)。

\labelenumiii enumerate 环境在列表嵌套中处于第三层时使用的列表项标识(p.32)

\labelenumiv enumerate 环境在列表嵌套中处于第四层时使用的列表项标识(p.32)。

\labelitemi itemize 环境在列表嵌套中处于第一层时使用的列表项标识(p.32)。

\labelitemii itemize 环境在列表嵌套中处于第二层时使用的列表项标识(p.32)。

\labelitemiii itemize 环境在列表嵌套中处于第三层时使用的列表项标识(p.32)。

\labelitemiv itemize 环境在列表嵌套中处于第四层时使用的列表项标识(p.32)。

\labelsep 包含列表项标识的文本盒与列表项文字之间的间隔(p.33)。

\labelwidth 包含列表项标识的文本盒的宽度(p.34)。

\lambda 数学模式下的 λ 字符(p.48)。

\Lambda 数学模式下的 Λ 字符(p.49)。

\land 数学模式下的∧（逻辑与）字符(p.50)。

\langle 数学模式下的⟨字符(p.51)。

\large, \Large, \LARGE 切换到三种比\normalsize 命令所代表的字号依次增大的字号(p.38)。

\LaTeX 产生 LaTeX 的标志 LaTeX(p.19)。

\lbrace 数学模式下的{符号(p.51)。

\lbrack 数学模式下的[符号(p.51)。

\lceil 数学模式下的⌈符号(p.51)。

\ldots 产生放置在当前行基线上的三个圆点…。比照\cdots 命令(p.29)。

\le 数学模式下的≤符号(p.49)。

\leadsto 数学模式下的⇝符号(p.51)。

\left* 用于数学模式，必须与\right*成对出现，其中*为定界符（也可以使用别的定界符） 圆点(.)用作表示空定界符(p.56)。

\leftarrow 数学模式下的←符号(p.51)。

\Leftarrow 数学模式下的⇐符号(p.51)。

\lefteqn{*formula*} 用于 eqnarray 环境，将一个太长得公式断为若干行(p.59)。

\leftharpoondown 数学模式下的↽符号(p.51)。

\leftharpoonup 数学模式下的↼符号(p.51)。

\leftmargin 用于列表环境，表示从列表项内容放置位置的左边界到整个列表环境的左边界之间的水平距离，可以使用\leftmargini, ..., \leftmarginvi 等 6 个命令将该距离的值依次设置为 1 至 6 档(p.34)。

\leftrightarrow 数学模式下的↔符号(p.51)。

\Leftrightarrow 数学模式下的⇔符号(p.51)。

\leq 数学模式下的≤符号(p.49)。

\lfloor 数学模式下的⌊符号(p.51)。

\lg 用于数学模式，代表 lg 函数名字符串(p.53)。

\lgroup 大左圆括号定界符(p.51)。

\lhd 数学模式下的⊲符号(p.50)。

\lim 用于数学模式，代表 lim 函数名字符串(p.53)。

\liminf 用于数学模式，代表 lim inf 函数名字符串(p.53)。

\limits 用于数学模式，修饰求和符号的上下标位置(p.54)。

\limsup 用于数学模式，代表 lim sup 函数名字符串(p.53)。

\line(*x*,*y*){*len*} 用于图形环境中的\put 命令，从\put 命令所带参数表示的起点绘制一条直线，x 轴上投影线长为 len，斜率为 y/x 。如果 x=0 且 y=1，表示直线与 y 轴平行，此时 len 表示直线的实际长度(p.93)。

\linebreak[*number*] 使当前行就此断开并对刚刚终止的一行进行调整，可比照\newline 命令(p.14)。

\linespread{*factor*} 改变行间距的值，factor 代表行间距放大的比例(p.39)。

\linethickness{*len*} 用于图形环境，设置所有直线的线宽(p.93)。

\linewidth 文本段落中当前行的行宽。

\listoffigures 开始一个带标题的图形列表清单(p.27)。

\listoftables 开始一个带标题的表的列表清单(p.27)。

\listparindent 用于列表环境，表示列表项内容中第一自然段后面的每一段额外缩进的距离(p.34)。

\ll 数学模式下的≪字符(p.49)。

\lmoustache 大的类似于积分符号的左定界符(p.51)。

\ln 用于数学模式，代表 ln 函数名字符串(p.53)。

\lnot 数学模式下的¬（逻辑非）字符(p.52)。

\LoadClass[*options*]{*class-name*}[*date*] 在类或包文件中装入别的类文件(p.110)。

\LoadClassWithOptions{*class-name*}[*date*] 在类或包文件中使用可选项装入别的类文件(p.110)。

\log 用于数学模式，代表 log 函数名字符串(p.53)。

\longleftarrow 数学模式下的⟵字符(p.51)。

\Longleftarrow 数学模式下的⟸字符(p.51)。

\longleftrightarrow 数学模式下的⟷字符(p.51)。

\Longleftrightarrow 数学模式下的⟺字符(p.51)。

\longmapsto 数学模式下的⟼字符(p.51)。

\longrightarrow 数学模式下的⟶字符(p.51)。

\Longrightarrow 数学模式下的⟹字符(p.51)。

\lor 数学模式下的∨（逻辑或）字符(p.50)。

\lowercase {*text*} 将text中的字母全部转换成小写，但不能转换某些特殊字符（如\AE或\AA）(p.120)。

\lq 左引号 ‘。

\makebox[*width*][*position*]{*text*} 创建容纳 text 文本的盒子，盒子的宽度为 width。文本的位置由 position 确定，有关位置的含义参见本附录中的 position 条目(p.41)。

\makebox(*width,height*)[*position*]{*text*} 类似于上一条命令，只是盒子的长度和宽度分别由 width 和 height 指定(p.91)。

\makeglossary 使得\glossaryentry 命令可以将术语写入术语表文件（.glo 文件）。

\makeindex 使得\indexentry 命令可以将索引项写入索引表文件（.idx 文件）(p.45)。

\MakeLowercase {*text*} 将text中的字母全部转换成小写，包括特殊字符（如\AE或\AA）(p.120)。

\maketitle 创建文档标题，题目和作者分别用\title 命令和\author 命令提供，另外可以用可选命令\date 提供文档日期(p.13)。

\MakeUppercase {*text*} 将text中的字母全部转换成大写，包括某些特殊字符（如\ae或\aa）(p.120)。

\mapsto 数学模式下的↦字符(p.51)。

\marginpar[*left*]{*right*} 将 text 放入边注区作为注释文本(p.28)。

\marginparpush 两个边注区注释文本之间的最小垂直间距(p.43)。

\marginparsep 页面边界与边注区注释文本之间的水平间距(p.43)。

\marginparwidth 边注区注释文本的宽度(p.43)。

\markboth{*left head*}{*right head*} 用于 headings 或 myheadings 页面风格，将左页页眉和右页页眉分别定义为 left head 和 right head(p.14)。

\markright{*right head*} 用于 headings 或 myheadings 页面风格，将右页页眉定义为 right head，面左页页眉保持不变(p.14)。

\mathbf{*text*} 将公式中的 text 内容用数学黑体输出(p.59)。

\mathit{*text*} 将公式中的 text 内容用数学斜体输出(p.59)。

\mathnormal{*text*} 将公式中的 text 内容用标准数学字体输出(p.59)。

\mathrm{*text*} 将公式中的 text 内容用数学 roman 字体输出(p.59)。

\mathsf{*text*} 将公式中的 text 内容用数学模式的 sans serif 字体输出(p.59)。

\mathtt{*text*} 将公式中的 text 内容用数学模式的 typewriter 字体输出(p.59)。

\mathversion{*bold*} 将公式中的 bold 内容加黑(p.59)。
\max 用于数学模式，代表 max 函数名字符串(p.53)。
\mbox{*text*} 将 text 文本放入一个宽度仅能容纳 text 的水平盒子(p.15)。
\mdseries 将文本字体切换为中等浓度(缺省值，与\bfseries 相反)。同\textmd 命令(p.37)。
\medskip 标准中等垂直间距(p.41)
\medskipamount \medskip 的缺省长度(p.41)。
\mho 数学模式下的℧字符(p.52)。
\mid 数学模式下的∣字符(p.49)。
\min 用于数学模式，代表 min 函数名字符串(p.53)。
\mit 切换至数学斜体字(math italic)。
\models 数学模式下的⊨字符(p.49)。
\month 当前月份。
\mp 数学模式下的∓字符(p.50)。
\mu 数学模式下的μ字符(p.48)。
\multicolumn{*num*}{*col*}{*text*} 用于表格环境，将文本 text 放入编号为 num 的列中，放置位置由格式参数 col 确定(p.100)。
\multiput(*x*, *y*) (*dx*, *dy*){*number*}{*element*} 相当于下面的命令序列：

```
\put(x, y){elememt}
\put(x+dx, y+dy){elememt}
......
\put(x+(number-1)*dx, y+(number-1)*dy){elememt}
```

(p.90)
\nabla 数学模式下的∇字符(p.52)。
\natural 数学模式下的♮字符(p.52)。
\ne 数学模式下的≠字符(p.49)。
\nearrow 数学模式下的↗字符(p.51)。
\NeedsTeXFormat{LaTeX2e}{*date other-information*} 类或包文件声明所使用的 LaTeX2e 基础类包文件的版本及其他信息(p.109)。
\neg 数学模式下的¬字符(p.52)。
\neq 数学模式下的≠字符(p.49)。
\newcommand{*cmd*}[*args*][*default*]{*definition*} 将新的命令 cmd 定义为 definition，args 说明 cmd 命令所需要的参数数量，缺省为 0；default 表示 cmd 命令的第一个参数是可选参数，且其缺省值为 default(p.23)。
\newcounter{*foo*}[*counter*] 定义新的计数器 foo 并将其值初始化为 0。可选的参数 counter 为另一个计数器，每当 counter 的值增加时，foo 计数器就将被复位(p.17)。
\newenvironment{*nam*}[*args*][*default*]{*begdef*}{*enddef*} 定义一个新的环境 nam，可选参数 args 表示它的参数个数。begdef 和 enddef 是两条命令参数，当进入环境 nam 时将执行 begdef，退出环境 nam 时将执行 enddef(p.23)。
\newfont{*cmd*}{*font_name*} 定义一个控制序列 cmd，cmd 将选择字体 font_name(p.24)。
\newlength{*gnat*} 将 \gnat 设置为代表 0 英寸的长度。参见\setlength, \addtolength 和

\settowidth 等命令(p.25)

\newline 立即终止当前行，被终止行不拉伸以占满一行。比照\linebreak 命令(p.14)。

\newpage 在当前位置结束本页。比照\clearpage 命令(p.15)。

\newsavebox{*cmd*} 声明一个新的同\savebox 合作的命令 cmd(p.41)。

\newtheorem{*name*}[*counter*]{*text*}[*section*] 只能在导言区出现，定义一个新的定理环境 name(p.24, 61)。

\ni 数学模式下的∋字符(p.49)。

\nofiles 禁止使用辅助文件（如.idx 文件和.toc 文件等）(p.27)。

\noindent 禁止段落首行缩进(p.16)。

\nolinebreak[*number*] 避免在当前位置换行。比照\linebreak 命令(p.15)。

\nonumber 用于 eqnarray 环境，禁止对公式编号(p.48)。

\nopagebreak[*number*] 避免在当前位置换页。比照\linebreak 命令(p.16)。

\normalcolor 将前景颜色切换到正常颜色。所谓正常颜色，是指文档导言区结束时被激活的前景颜色(p.126)。

\normalfont 将文本字型切换为主文档标准字型。同\textnormal 命令(p.37)。

\normalmarginpar 边注区注释的缺省放置方式。比照\reversemarginpar 命令。

\normalsize 文档的缺省字体尺寸(p.38)。

\not 用于数学模式，在关系运算符上面画一条斜线，如\not=代表≠字符(p.52)。

\notin 数学模式下的∉字符(p.49)

\nu 数学模式下的ν字符(p.48)。

\nwarrow 数学模式下的↖字符(p.51)。

\o 字符 ø(p.30)。

\O 字符 Ø(p.30)

\obeycr 将内置的回车符作为行终止符对待。

\oddsidemargin 页面左边到正文正常左边界的距离(p.43)。

\odot 数学模式下的⊙字符(p.50)。

\oe 字符 œ(p.30)。

\OE Œ 字符(p.30)。

\oint 数学模式下的∮字符(p.50)。

\omega 数学模式下的ω字符(p.48)。

\Omega 数学模式下的Ω字符(p.49)。

\ominus 数学模式下的⊖字符(p.50)。

\onecolumn 设置单栏排版方式（缺省情况）。比照\twocolumn 命令(p.12)。

\opening{*text*} 用于书信文档格式，声明正文开始。

\oplus 数学模式下的⊕字符(p.50)。

\OptionNotUsed 用于类或包文件中的可选项代码，将用户指定的当前可选项添加到“未使用可选项”列表(p.115)。

\oslash 数学模式下的⊘字符(p.50)。

\otimes 数学模式下的⊗字符(p.50)。

\oval(*x*,*y*) [*opt*] 作为\put 命令的参数使用，绘制一个宽 x 单位、高 y 单位的圆角矩形，可选参数 opt 控制不完整的圆角矩形的绘制(p.94)。

\overbrace{*text*}[^{*upper-note*}] 用于数学模式，在 text 上面放置一开口朝下的花括号，upper-note 为放在花括号上方的说明文字，大小为上标大小(p.54)。

\overline{*text*} 用于数学模式，在 text 所代表的文字上面画一条直线(p.54)。

\owns 数学模式下的∋字符(p.49)。

\P ¶符号(p.52)。

\PackageError {*package-name*} {*error-text*} {*help-text*} 用于单独的类或包文件，当包文件package-name出错时负责向用户报告或提供相关错误信息(p.118)。

\PackageInfo {*package-name*} {*info-text*} 同\PackageWarning命令相似，只是将info-text信息及警告发生行行号不显示，而是写入log文件(p.119)。

\PackageWarning {*package-name*} {*warning-text*} 用于单独的类或包文件，当包文件package-name出错时在屏幕上产生warning-text警告信息及警告发生处的行号，但没有帮助信息(p.119)。

\PackageWarningNoLine {*package-name*} {*warning-text*} 同\PackageWarning命令相似，只是不显示警告发生处的行号(p.119)。

\pagebreak[*number*] 强迫换页。比照\linebreak 命令(p.15)。

\pagecolor *col_spec* 设置当前及后续页面的背景颜色(p.125)。

\pagenumbering{*num_style*} num_style 参数指定页码格式(p.13)。

\pageref{*key*} 由参数 key 对应的\label{key}命令所在页面的页码(p.18)。

\pagestyle{*style*} 指定页面头部（顶部）和尾部的排版特征(p.13)。

\paperheight 通常用于文档类文件，设置纸张的高度(p.43)。

\paperwidth 通常用于文档类文件，设置纸张的宽度(p.43)。

\par 与一个空行的效果相同，使用它常常只是为了使文档中的命令或环境的定义更易于阅读理解(p.16)。

\paragraph[*toctitle*]{*text*} 开始自动设标题和编号的新的段落。可选参数 toctitle 如果与 text 内容不同，则将作为目录表中该段落相应条目的文字(p.26)。

\paragraph*{*text*} 开始新的段落并打印标题，但是不带编号，也不在目录表中产生相应条目(p.26)。

\parallel 数学模式下的字符∥(p.49)。

\parbox[*position*][*height*][*inner-pos*]{*width*}{*text*} 产生段落模式下的文本盒(p.42)。

\parindent 段落开始处的水平缩进量(p.40)。

\parsep 同一列表项内部不同段落之间额外的垂直间隔(p.34)。

\parskip 两个普通段落之间额外的垂直间隔(p.34)。

\part[*toctitle*]{*text*} 开始自动设标题和编号的新的部分。可选参数 toctitle 如果与 text 内容不同，则将作为目录表中该部分相应条目的文字(p.26)。

\part*{*text*} 开始新的部分并打印标题，但是不带编号，也不在目录表中产生相应条目。

\partial 数学模式下的∂字符(p.52)。

\partopsep 如果通用列表环境中对每个列表项开始新的一段，则该命令表示第一个列表项段落前面额外的垂直间隔(p.34)。

\PassOptionsToClass{*option*}{*class*} 将装入可选项option传递到另一个类文件class(p.111)。

\PassOptionsToPackage{*option*}{*package*} 将装入可选项option传递到另一个包文件package(p.111)。

\perp 数学模式下的⊥字符(p.49)。

\phi 数学模式下的ϕ字符(p.48)。

\Phi 数学模式下的Φ字符(p.49)。

\pi 数学模式下的π字符(p.48)。

\Pi 数学模式下的Π字符(p.49)。

\pm 数学模式下的 ± 字符(p.50)。

\pmod{*modulus*} 用于数学模式，是带括号的求模表达式(p.71)。

\poptabs 取消前面\pushtabs 命令的作用，恢复以前的制表位设置(p.21)。

positions 用一个字母代表一种位置：t 代表顶部(top)，b 代表底部(bottom)，h 代表就地(here)，l 代表左边(left)，c 代表中间(center)，r 代表右边(right)；图形环境中 p 代表换新页；表格环境中 p 代表 parbox。

\pounds £ 符号(p.52)。

\Pr 用于数学模式，代表 Pr 函数名字符串(p.53)。

\prec 数学模式下的≺字符(p.49)。

\preceq 数学模式下的⪯字符(p.49)。

\prime 数学模式下的′字符(p.52)。

\ProcessOptions 用于类或包文件，执行每个使用的可选项在声明时设定的处理代码(p.116)。

\prod 数学模式下的∏字符(p.50)。

\propto 数学模式下的∝字符(p.49)。

\protect 允许在@表达式、分节命令或\caption 命令的参数中使用“危险”命令(p.119)。

\providecommand 同 newcommand 命令，惟一的区别在于如果新定义的命令名称已经存在，LaTeX 将不出错，而只是忽略该条命令(p.23)。

\ProvidesClass{*class-name*}[*date other-information*] 类文件声明自己提供的类名称及主要功能特性等信息(p.114)。

\ProvidesFile {*file-name*} [*release-info*] 同ProvidesClass和ProvidesPackage命令相似，只是file-name必须为文件全名（即后缀不能省略）。这条命令可以用来声明除了主类和包文件以外的文件(p.114)。

\ProvidesPackage{*package*}[*date other-information*] 包文件声明自己提供的包名称及主要功能特性等信息(p.114)。

\ps 用于书信文档，允许在\closing 命令之后添加附加的文字(p.112)。

\psi 数学模式下的ψ字符(p.48)。

\Psi 数学模式下的Ψ字符(p.49)

\pushtabs 用于制表位环境，使制表位设置入栈。可用\poptabs 命令恢复(p.21)。

\put(*x*,*y*){*element*} 基本的作图命令，(x,y)是参考点坐标，其含义随 stuff 参数的不同而不同 Stuff 可以是放进\mbox 命令中的任何东西(p.90)。

\qbezier{*number*} (x_1, y_1)(x_2, y_2)(x_3, y_3) 在(x_1,y_1)点到(x_3,y_3)点之间绘制一条bézier曲线，(x_2,y_2)为曲线控制点。number参数表示曲线所经过的路段数目(p.95)。

\qquad 留一个较大距离的空白（差不多相当于按下 Tab 键所跳过的距离）(p.57)

\quad 同\qquad 命令(p.57)。

\raggedbottom 页面采用本页文档的自然高度排版。

\raggedleft 声明所有后续文本都与右边界平齐，比照\befin{flushright}命令。

\raggedright 声明所有后续文本都与左边界平齐，比照\befin{flushleft}命令。

\raisebox{*distance*}[*extend-above*][*extend-below*]{*text*} 将文本 text 升高 distance 距离（distance 可取负值）。extend-above 表示让文本超出顶线的上面多少距离，extend-below 表示让文本超出底线的下面多少距离(p.42)。

\rangle 数学模式下的⟩字符(p.51)。

\rbrace 数学模式下的}字符(p.51)。

\rbrack 数学模式下的]字符(p.51)。

\rceil 数学模式下的⌉字符(p.51)。

\Re 数学模式下的ℜ字符(p.52)。

\ref{*key*} key 参数对应的\label{key}命令出现的地方所属的分节或公式的编号(p.18)。

\reflectbox{*text*} 得到 text 区域的水平倒影，相当于区域中心为基点旋转 180 度(p.123)。

\refstepcounter{*counter*} 除了完成 \stepcounter 命令的功能以外，还将 \ref 的当前值作为\thecounter 的结果(p.17)。

\renewcommand{*cmd*}[*args*][*default*]{*definition*} 用 definition 将已有的命令 cmd 重新定义。参见\newcommand 命令(p.23)。

\renewenvironment{*nam*}[*args*]{*begdef*}{*enddef*} 重新定义已有的环境 nam。参见\newenvironment 命令(p.23)。

RequirePackage[*options*]{*package*}[*date*] 在一个类或包文件中装入别的包文件(p.109)。

\RequirePackageWithOptions{*package*}[*date*] 在一个类或包文件中使用可选项装入别的包文件(p.110)。

\resizebox{*h_length*}{*v_length*}{*text*} 将插入图像调整到指定的水平和垂直尺寸。其中 h_length 代表水平宽度，v_length 代表垂直高度，二者都可以使用!作为参数值，此时代表该方向按照 1:1 比例确定图像的原始尺寸(p.123)。

\restorecr 取消\obeycr 命令的作用效果（即将回车字符的作用解释为产生空格）。

\reversemarginpar 将边注区注释文本放到相反方向去（如在奇数页中将注释放到左边注区）(p.28)。

\rfloor 数学模式下的⌋字符(p.51)。

\rgroup 大的右圆括号定界符(p.51)。

\rhd 数学模式下的▷字符(p.50)

\rho 数学模式下的ρ字符(p.48)

\right* 用于数学模式，必须与\left*成对出现，其中*为定界符（也可以使用别的定界符）。圆点(.)用作表示空定界符(p.56)。

\rightarrow 数学模式下的→字符(p.51)。

\Rightarrow 数学模式下的⇒字符(p.51)。

\rightharpoondown 数学模式下的⇁字符(p.51)。

\rightharpoonup 数学模式下的↼字符(p.51)。

\rightleftharpoons 数学模式下的⇌字符(p.51)。

\rightmargin 用于列表环境，表示列表项内容的右边界距列表环境右边界的间隔(p.34)。

\rm 切换到 roman 字型。

\rmfamily 切换到 roman 字型。同\textrm 命令(p.37)。

\rmoustache 大的类似于反积分符号的右定界符(p.51)。

\roman{*counter*} 以小写罗马数字的方式打印计数器 counter(p.17)。

\Roman{*counter*} 以大写罗马数字的方式打印计数器 counter(p.17)。

\rotatebox{*angle*}{*text*} 将 text 以盒子基线的左端为基点，沿逆时针方向旋转 angle 度(p.123)

\rq 右引号'。

\rule[*raise-height*]{*width*}{*thickness*} 产生水平标尺线，raise-height 指定标尺线升高多少，width 指定标尺的水平长度，thickness 指定标尺的垂直厚度(p.42)。

\S §字符(p.52)

\savebox{*cmd*}[*width*][*pos*]{*text*} 与\makebox 极其相似（参见相关内容），但是将盒子的定义存于 cmd 中。使用盒子时可以用\usebox{cmd}命令(p.41)。

\sbox{*cmd*}{*text*} 将文本 text 存入 cmd 代表的盒子中。参见上一条命令(p.41)。

\sc 切换到 caps 和 small caps 字体。

\scalebox{*h_scale*}[*v_scale*]{*text*} 将矩形盒子水平和垂直缩放。其中参数 h_scale 代表水平方向的放大倍数；可选参数 v_scale 代表垂直方向的放大倍数，缺省时与 h_scale 相同(p.123)。

\scriptsize 切换到 subscript 字号（下标字号）大小(p.38)。

\scriptscriptstyle 将输出字号切换成二级下标或上标字号大小(p.60)。

\scriptstyle 将输出字号切换成下标或上标字号大小(p.60)。

\scshape 将文本字体切换为小体大写字母(small cap)。同\textsc 命令(p.37)。

\searrow 数学模式下的↘字符(p.51)。

\sec 用于数学模式，代表 sec 函数名字符串(p.53)。

\section[*toctitle*]{*text*} 开始自动设标题和编号的新的分节。可选参数 toctitle 如果与 text 内容不同，则将作为目录表中该分节相应条目的文字(p.26)。

\section*{*text*} 开始新的分节并打印标题，但是不带编号，也不在目录表中产生相应条目(p.26)

\selectfont \fontencoding, \fontfamily, \fontseries, \fontshape, \fontsize 等几条字体改变命令在遇到这条命令之前并不起作用(p.39)。

\setcounter{*counter*}{*value*} 重新将计数器 counter 的值设置为 value (p.17)。

\setlength{*gnat*}{*length*} 将长度命令\gnat 的值设置为 length 大小 。参见\addtolength, \newlength 和\settowidth 等命令(p.25)。

\setminus 数学模式下的\字符(p.50)。

\settodepth{*gnat*}{*text*} 将长度命令\gnat 的值设置为同文本参数 text 的深度相等(p.25)。

\settoheight{*gnat*}{*text*} 将长度命令\gnat 的值设置得同文本 text 的高度相等(p.25)。

\settowidth{*gnat*}{*text*} 将长度命令\gnat 的值设置为文本 text 的宽度(p.26)。

\sf 切换到 sans serif 字体。

\sffamily 将文本字型切换为 sans serif 字体。同\textsf 命令(p.37)。

\sharp 数学模式下的#字符(p.52)。

\shortstack[pos]{x\\yy\\zzz} 将参数中\\分隔的各个部分用一列多行的方式堆叠。pos 表示对齐方式(p.92)。

\sigma 数学模式下的σ字符(p.48)。

\Sigma 数学模式下的Σ字符(p.49)。

\signature{*text*} 用于书信文档类型，将 text 声明为发信人署名。

\sim 数学模式下的~字符(p.49)。

\simeq 数学模式下的≃字符(p.49)。

\sin 用于数学模式，代表 sin 函数名字符串(p.53)。

\sinh 用于数学模式，代表 sinh 函数名字符串(p.53)。

\sl 切换到小斜体字体。

\sloppy 将断行的算法放宽，允许行中单词间的空白适当拉大或缩小。文档的缺省设置为\fussy (p.15)。

\slshape 将文本字型切换为小斜体（slant，与\upshape 相反）。同\textsl 命令(p.37)

\small 切换到比\normalsize 设定的正常字号稍小的字号(p.38)。

\smallint 数学模式下的∫字符。

\smallskip 比标准垂直间隔距离稍小的间隔(p.41)。

\smallskipamount \smallskip 命令所代表的缺省长度(p.41)。

\smile 数学模式下的⌣字符(p.49)。

\spadesuit 数学模式下的♠字符(p.52)。

\sqcap 数学模式下的⊓字符(p.50)。

\sqcup 数学模式下的⊔字符(p.50)。

\sqrt[*n*]{*arg*} arg 的 n 次开方符号(p.55)。

\sqsubset 数学模式下的⊏字符(p.49)。

\sqsubseteq 数学模式下的⊑字符(p.49)。

\sqsupset 数学模式下的⊐字符(p.49)。

\sqsupseteq 数学模式下的⊒字符(p.49)。

\ss ß 字符(p.29)。

\stackrel{*top-text*}{*bottom-text*} 用于数学模式，将 top-text 文本放到 bottom-text 文本的上方(p.53)。

\star 数学模式下的⋆字符(p.50)。

\stepcounter{*counter*} 将计数器 counter 的值加 1，同时复位所有的辅助计数器(p.17)。

\stop 文本结束，如果没有错误就进行排版。

\stretch 和\pagebreak 命令相连使用可以对一页的最后一行文本进行排版，或者使文本在垂直方向上向页面中央居中对齐(p.40)。

\subparagraph[*toctitle*]{*text*} 开始自动设标题和编号的新的子段落。可选参数 toctitle 如果与 text 内容不同，则将作为目录表中该子段落相应条目的文字(p.26)。

\subparagraph*{*text*} 开始新的子段落并打印标题，但是不带编号，也不在目录表中产生相应条目(p.26)。

\subsection[*toctitle*]{*text*}, \subsubsection[*toctitle*]{*text*} 开始自动设标题和编号的新的子分节。可选参数 toctitle 如果与 text 内容不同，则将作为目录表中该子分节相应条目的文字(p.26)。

\subsection*{*text*}, \subsubsection*{*text*} 开始新的子分节，但是不带编号，也不在目录表中产生相应条目(p.26)。

\subset 数学模式下的⊂字符(p.49)。

\subseteq 数学模式下的⊆字符(p.49)。

\succ 数学模式下的≻字符(p.49)。

\succeq 数学模式下的≽字符(p.49)。

\sum 数学模式下的∑字符(p.50)。

\sup 用于数学模式，代表 sup 函数名字符串(p.53)。

\suppressfloats[*placement specifier*] 确保在当前页中由 placement specifier 指定的区域不再放置可浮动体(p.104)。

\supset 数学模式下的⊃字符(p.49)。

\supseteq 数学模式下的⊇字符(p.49)。

\surd 数学模式下的√字符(p.52)。

\swarrow 数学模式下的↙字符(p.51)。

\symbol{*char-code*} 产生在当前字体中字符代码为 char-code 的符号或字形(p.6)。

\t 输出与后一个字符相连的连音符号(p.30)。

\tabbingsep 制表位环境下用\'命令向左跳过的距离(p.22)。

\tabcolsep 用于表格环境中，表示两列间空格宽度的一半(p.101)。

\tableofcontents 产生一个目录表。要求上次运行 LaTeX 时必须已经产生了一个.toc 文件(p.26)。

\tabularnewline 用于 tabular 或 array 环境，强制一表格行的结束(p.99)。

\tan 用于数学模式，代表 tan 函数名字符串(p.53)

\tanh 用于数学模式，代表 tanh 函数名字符串(p.53)。

\tau 数学模式下的字符 τ(p.48)。

\TeX 产生 TeX 的标志 TEX。
\textbf 将文本字体加黑 同\bfseries 命令(p.37)。
\textcolor *col_spec*{*text*} 将 text 参数所代表的文本以 col_spec 颜色输出(p.126)。
\textfloatsep 单栏页面中顶部或底部可浮动体距页面中文本之间的间隔(p.104)。
\textfraction 文本页面中文字必须占用的最小比例(p.104)。
\textheight 页面中用于排版文档的区域的正常垂直尺寸(p.16)。
\textit 将文本字型切换为斜体(italic)。同\itshape 命令(p.37)。
\textmd 将文本字体切换为中等浓度（缺省值，与\textbf 相反）。同\mdseries 命令(p.37)
\textnormal 将文本字型切换为主文档标准字型。同\normalfont 命令(p.37)。
\textrm 将文本字型切换为 roman 字体。同\rmfamily 命令(p.37)。
\textsc 将文本字体切换为小体大写字母(small cap)。同\scshape 命令(p.37)。
\textsf 将文本字型切换为 sans serif 字体。同\sffamily 命令(p.37)。
\textsl 将文本字型切换为小斜体（slant，与\textup 相反）。同\slshape 命令(p.37)。
\textstyle 用于数学模式，将输出字号切换到文本模式字体大小(p.60)。
\texttt 将文本字型切换为 typewriter 字体。同\ttfamily 命令(p.37)。
\textup 将文本字型切换为正体（缺省值，与\textsl 相反）。同\upshape 命令(p.37)。
\textwidth 页面中用于排版文档的区域的正常水平尺寸(p.43)。
\thanks{*text*} 用于\maketitle 命令，在一个作者名的后面加上一段致谢的脚注(p.13)。
\theta 数学模式下的θ字符(p.48)。
\Theta 数学模式下的Θ字符(p.49)。
\thicklines 用于图形环境，是一种可选的直线线宽。参见\linethickness 命令(p.93)。
\thickspace 表示较大的间隔距离(p.82)。
\thinlines 用于图形环境，是缺省的直线线宽。参见\thicklines 命令(p.93)。
\thinspace 表示较小的间隔距离(p.82)。
\thispagestyle{*style*} 指定仅适用于当前页的页面类型，即临时覆盖\pagestyle 命令所设置的页面类型。参见\pagestyle 命令(p.13)。
\tilde 用于数学模式，在字母上方产生波浪线（如 $\tilde{n}$）(p.48)。
\times 数学模式下的×字符(p.50)。
\tiny 切换到一种极小的字体(p.38)
\title{*text*} 用于\maketitle 命令，声明文档的标题(p.13)。
\to 数学模式下的→字符(p.51)。
\today 产生计算机中的当前日期。
\top 数学模式下的⊤字符(p.52)。
\topfigrule 在页面顶部浮动图表之后被执行的命令(p.105)。
\topfraction 单分栏页面中顶部可以用来放置浮动体的部分所能占的最大比例(p.104)。
\topmargin TeX 页面顶部（从整个页面的顶部下移 1 英寸处）到页面头部内容间的距离(p.43)。
\topsep 第一个列表项前面及最后一个列表项后面所额外增加的垂直空白(p.34)。

\topskip 页体（页面中所有用于文档排版区域）顶部距第一行文字底部的最小间隔。

\totalheight 文本字符的总高度(\height + \depth)(p.25)。

\triangle 数学模式下的△字符(p.52)。

\triangleleft 数学模式下的◁字符(p.50)

\triangleright 数学模式下的▷字符(p.50)

\tt 切换到 typewriter 字体。

\ttfamily 将文本字型切换为 typewriter 字体。同\texttt 命令(p.37)。

\twocolumn[*text*] 声明双栏页面，可选参数 text 为按整页宽度（不分栏）放置的标题(p.12)。

\typein[*cs*]{*text*} 将文本 text 显示在屏幕上并等待你输入将在此位置插入的内容。可选的控制序列参数\cs 可以用来代表你所输入的东西，以便后面重复使用。

\typeout{*text*} 将文本 text 显示在屏幕上，同时写入.lis 文件中(p.111)。

\u 产生短弱发音符号（如 ă）。比照\breve 命令(p.30)。

\unboldmath 用于数学模式外部，表示非粗体的数学斜体和数学符号。

\underbrace{*text*}[_{*under-note*}] 用于数学模式，在文本 text 下方放置开口朝上的花括号，under-note 作为花括号下面下标大小的说明文字(p.54)。

\underline{*text*} 不论是否数学模式均可使用，使文本 text 带下画线(p.54)。

\unitlength 用于图形模式，表示坐标单位的大小(p.89)。

\unlhd 数学模式下的⊴字符(p.50)。

\unrhd 数学模式下的⊵字符(p.50)。

\uparrow 数学模式下的↑字符(p.51)

\Uparrow 数学模式下的⇑字符(p.51)

\updownarrow 数学模式下的↕字符(p.51)

\Updownarrow 数学模式下的⇕字符(p.51)

\uplus 数学模式下的⊎字符(p.50)。

\uppercase {*text*} 将text中的字母全部转换成大写，但不能转换某些特殊字符（如\ae 或\aa）(p.120)

\upshape 将文本字型切换为正体（缺省值，与\slshape 相反）。同\textup 命令(p.37)。

\upsilon 数学模式下的υ字符(p.48)

\Upsilon 数学模式下的Υ字符(p.49)

\usebox{*cmd*} 使用 cmd 中存放的盒子的定义(p.42)。

\usecounter{*counter*} 用于列表环境，使用计数器 counter 对列表项进行计数(p.17)。

\usefont{*enc*}{*family*}{*series*}{*shape*} 相当于使用相应的参数依次调用\fontencoding, \fontfamily, \fontseries 和\fontshape 命令，最后调用\selectfont 命令(p.39)。

\usepackage[*options*]{*package*} 装入包文件 package (p.7)。

\v 产生诸如 è 的发音符号(p.30)。

\value{*counter*} 用数值表示的计数器 counter 的当前值(p.18)。

\varepsilon 数学模式下的ε字符(p.48)

\varphi　数学模式下的φ字符(p.48)。

\varpi　数学模式下的ϖ字符(p.48)。

\varrho　数学模式下的ϱ字符(p.48)

\varsigma　数学模式下的ς字符(p.48)

\vartheta　数学模式下的ϑ字符(p.48)。

\vdash　数学模式下的$\vdash$字符(p.49)

\vdots　数学模式下的$\vdots$字符(p.52)。

\vec　用于数学模式，在字母上方画一右向箭头（可表示矢量）(p.53)

\vector(*x*,*y*){*len*}　用于图形环境中的\put 命令，从\put 参数代表的点开始绘制长度为 len、斜率为 y/x 的直线箭头(p.93)。

\vee　数学模式下的 v 字符(p.50)。

\verb /*text*/　为文本 text 创建一个局部的字面(verbatim)环境，字体采用 typewriter 字体。注意 text 参数不是放在一对花括号中，而是放在两个左斜杠之间，左斜杠并不在排出的 text 文本中出现(p.20)。

\verb* /*text*/　同\verb/text/命令相似，只是将 text 文本中出现的空格输出为 ␣ (p.20)。

\vert　数学模式下的 | 字符(p.51)。

\Vert　数学模式下的 ‖ 字符(p.51)。

\vfill　相当于命令\vspace{\fill}。比照\fill 命令(p.25)。

\vline　用于表格环境,画一条竖直线，其高度等于所在位置的行高(p.100)。

\vspace{*length*}　在垂直方向留下长度为 length 的空白(p.40)。

\vspace*{*length*}　类似于命令\vspace{length}，只是留下的空白即使处于页首或页尾位置也予以保留而不删除(p.40)。

\wedge　数学模式下的 ∧ 字符(p.50)

\widehat{*arg*}　用于数学模式，在文本 arg 上方放置一个拉长的 ∧ 状符号(p.53)。

\widetilde{*arg*}　用于数学模式，在文本 arg 上方放置一个拉长的波浪线(p.53)。

\width　文本字符宽度(p.25)。

\wp　数学模式下的草写体 p($\wp$)(p.52)。

\wr　数学模式下的≀(p.50)。

\xi　数学模式下的ξ字符(p.48)。

\Xi　数学模式下的Ξ字符(p.49)。

\year　当前年份（公元后）

\zeta　数学模式下的ζ字符(p.48)。